Departamento de Ingeniería de Sistemas y Automática

TRABAJO FIN DE GRADO

Grado en Ingeniería Electrónica, Industrial y Automática

Onewebmicrodata: Araña web extractora de Metadatos de Microdata y Exif

Autor: Julián Caro Linares

Tutor: Juan Carlos González Víctores

Director: Santiago Morante Cendrero

Leganés, Marzo, 2014

Título: Onewebmicrodata: Araña web extractora de Metadatos de Microdata y Exif - 2014

Autor: Julián Caro Linares

Tutor: Juan Carlos González Víctores

Director: Santiago Morante Cendrero

EL TRIBUNAL

Presidente: Miguel Caballero Salichs Sánchez

Vocal: Julio Usaola García

Secretario: Juan Carlos Torres Zafra

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día 10 de Marzo de 2014 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de:

VOCAL

SECRETARIO PRESIDENTE

"Los límites de mi lenguaje son los límites de mi conocimiento."

Ludwig Wittgenstein

Agradecimientos

En primer lugar, quiero agradecer de forma especial a mi tutor Juan G. Victores la oportunidad que me ha brindado con este proyecto. Gracias a él he descubierto un campo de estudio que relaciona la linguística con la ingeniería de un modo apasionante. Sin su buen humor, talento y, sobre todo, su gran diligencia y ayuda, no hubiera podido terminar un proyecto que, hace cinco años, me hubiera considerado incapaz de realizar.

Quiero agradecer también a Adolfo, Arturo, Celia, Daniel, Elena, Javier, Marco y Roberto por esta gran aventura que hemos vivido juntos. Hemos pasado muchos momentos duros, muchas horas de trabajo agotadoras persiguiendo un sueño que siempre nos parecía demasiado lejano, pero cada risa obtenida a cambio ha hecho que merezca la pena. Gracias a vosotros he aprendido que la amistad y el trabajo en equipo consiguen cualquier cosa.

Ringrazio anche ai miei amici Charlotte, Eleonora, Ilaria, Maria e Nuno. Nell'anno dell'Erasmus che abbiamo trascorso insieme mi sono innamorato di Milano e, soprattutto, di voi. Grazie a voi ho capito che la vera amicizia supera paesi,confini e tempo. La vita passa in fretta e a volte non riusciamo a trovare tempo per ciò che è importante, ma sarò sempre qui per voi

Por último, dedico este proyecto a mi familia. A mi padre, a mi madre, y a mi hermana Estrella, así como a mis abuelos, tíos y primos, gracias por todo. Gracias por permitirme siempre ser quien soy, por mantenerme y alentarme en los momentos duros y enseñarme que un fracaso es solo una coma, y un logro siempre será solo un punto y aparte. Soy muy afortunado por teneros a mi lado.

Resumen

El presente proyecto trata sobre la creación y desarrollo de un programa tipo *araña web* o *crawler* cuyo objetivo sea la localización y descarga de imágenes descritas mediante metadatos del tipo *microdata* y *Exif*, así como la extracción de éstos. El programa realizado ha sido creado con la intención de proporcionar una herramienta automatizada de obtención de imágenes con información asociada. El fin último de esta herramienta es el de alimentar bases de datos de sistemas robóticos o agentes inteligentes. A lo largo del documento, el lector conocerá la historia y funcionamiento de la web semántica mediante la presentación de los distintos lenguajes, vocabularios y tecnologías que la componen. Conocerá a su vez el funcionamiento y aplicaciones, tanto actuales como futuras, de la *microdata*, y cómo su uso puede significar un paso adelante en el futuro desarrollo de la web actual.

Se explicará, además, en qué consiste un programa tipo *crawler* como el creado en este proyecto, desde su funcionamiento básico, a sus principales problemas y múltiples aplicaciones. Se describirá el programa realizado, desde sus objetivos, especificaciones de diseño y funcionamiento básico, a una descripción más exhaustiva de su arquitectura. Se expondrá además una serie de ensayos realizados para comprobar el comportamiento del programa junto con sus resultados.

En definitiva, el presente proyecto trata de ser un acercamiento a la tecnología de la web semántica, desarrollando un programa capaz de extraer y analizar la información de uno de sus lenguajes más usados, la *microdata*. La extracción de mayor información presente en la web abre un amplio abanico de futuros usos en las que distintas aplicaciones y máquinas puedan usar dicha información con múltiples propósitos.

Palabras clave: araña web, crawler, microdata, web semántica, WEB 3.0.

Abstract

The aim of this project is the creation and development of a *web spider* or *crawler* program, with the purpose of locating and downloading images described with metadata such as *microdata* and *Exif*, and the extraction of this data. The program has been developed with the intention of providing an automated tool for obtaining images with associated information. The ultimate goal of this tool is to feed databases of robotic systems or intelligent agents. Throughout this document, the reader will learn the semantic web's history and how it works, exhibiting the diferent languages, vocabularies and tecnologies that form it. The reader will also discover how *microdata* works, and how its use can mean a step ahead in the future development of the current web.

Furthermore, this document will show what a *crawler* is, from its basic functioning, to its main problems and multiple applications. The developed program will be explained, starting with its goals, design specifications and basic behavior, up to a description of its functional architecture. A set of tests conducted to check the correct running of the program will also be exhibited.

In conclusion, the present project is a first approach to the semantic web tecnology, developing a program able to extract and parse information from one of the semantic web's most employed languages: *microdata*. The use of this tecnology opens a great field of future uses in which different applications and machines could use the extracted information for multiple purposes.

Keywords: web spider, crawler, microdata, semantic web, WEB 3.0.

Índice general

Agradecimientos v

Resumen vii

Abstract ix

1. Introducción 1

1.1. Motivación del proyecto . . . 3

1.2. Objetivos específicos . . . 4

1.3. Estructura del documento . . . 5

2. Estado del arte 7

2.1. Web semántica . . . 7

2.1.1. Evolución de la web, el paso de la web de documentos a la web social, la transición hacia la web de datos . . . 8

2.1.2. Estructura de la web semántica . . . 10

2.1.3. Aplicaciones de la web semántica . . . 16

2.2. Microdata y HTML5: Acercando la web semántica al diseño web . . . 19

2.2.1. Schema.org: El esquema adoptado por los grandes buscadores . . . 20

2.2.2. Estructura de la microdata . . . 20

2.2.3. Herramientas y recursos de microdata actuales . . . 23

2.2.4. Ventajas y desventajas respecto a otros vocabularios . . . 24

2.2.5. Aplicaciones presentes y futuras . . . 25

2.3. Arañas web: Herramientas para la obtención de información en la web . . . 26

2.3.1. Descripción general y funcionamiento . . . 27

2.3.2. Aplicaciones actuales . . . 28

3. Descripción del programa *Onewebmicrodata* 31

3.1. Descripción general . . . 31

3.2. Especificaciones de diseño . . . 32
3.2.1. Objetivos generales y requisitos del sistema . . . 32
3.2.2. Lenguaje de programación utilizado . . . 33
3.2.3. Librerías externas utilizadas . . . 34
3.2.4. Estructura básica del programa . . . 36
3.3. Definición de la arquitectura . . . 37
3.3.1. Bloque 1: Descarga página web . . . 37
3.3.2. Bloque 2: Análisis del *HTML* . . . 40
3.3.3. Bloque 3: Extracción de código con microdata . . . 43
3.3.4. Bloque 4: Análisis y extracción de la microdata . . . 45
3.3.5. Bloque 5: Iteración a la siguiente página . . . 51
3.3.6. Bloque de control: main . . . 51

4. Ensayos y resultados **63**
4.1. Búsqueda en freefoto.com . . . 65
4.2. Búsqueda en dreamstime.com . . . 68
4.3. Búsqueda en istockphoto.com . . . 72
4.4. Datos Exif . . . 75
4.5. Un ejemplo de búsqueda múltiple . . . 77

5. Conclusiones **81**
5.1. Conclusiones sobre el programa desarrollado . . . 81
5.2. Desarrollos futuros . . . 84

A. Manual de uso **87**
A.1. Introducción . . . 87
A.2. Compilación y ejecución del programa . . . 87
A.3. Opciones del programa y configuración . . . 88
A.4. Un ejemplo de uso . . . 95
A.5. Recomendaciones y posibles problemas . . . 97

Bibliografía **99**

Índice de figuras

1.1. Arquitectura del sistema a alimentar con datos semánticos. Figura extraída del artículo *Towards Robot Imagination Throught Object Feature Inference*, 2013. 3

1.2. Ejemplo de imágenes de entrenamiento extraído del artículo *Towards Robot Imagination Throught Object Feature Inference*, 2013. Cada imagen tiene asociada información mediante palabras clave del tipo: rectángulo-azul-esquina-superior-izquierda. 4

2.1. Estructura de la web de documentos: la web se entiende como una red de documentos relacionados siendo éste su unidad más pequeña. 10

2.2. Ejemplo de la web de datos. Cada objeto representa un ente real o concepto abstracto relacionados entre sí. 11

2.3. Ejemplo de dos conjuntos de datos no relacionados entre sí. Las dos personas representadas trabajan para la misma empresa y ambas tienen el mismo modelo de teléfono móvil. Sin embargo ambos datos están duplicados y los datos de los dos conjuntos no se interrelacionan. 11

2.4. Ejemplo de web de datos usando la web semántica. 12

2.5. Tarta Semántica propuesta por Tim Berners-Lee, 2000 12

2.6. Ejemplo de ontología para un catálogo multimedia. 14

2.7. Nueva Tarta Semántica propuesta por el W3C, 2006 15

2.8. *The linked open data cloud diagram*, linkeddata.org, 2011 muestra los principales dominios de Internet que contienen información semántica y su relación entre ellos. Destaca el proyecto *DBpedia* como uno de los nodos más relacionados. 18

2.9. Resultados del ejemplo de código con microdata tipo *person* extraídos por la herramienta *Google Structured Data Testing Tool*. 23

2.10. Ejemplo de resultados enriquecidos, Google, 2013 25

2.11. Resumen de los resultados obtenidos por *Incapsula Inc*, 2013, relativos al tráfico humano y no humano en el año 2013. 26

2.12. Arquitectura general de un *web crawler*, Carlos Castillo, 2004. 28

3.1. Estructura general del comportamiento del programa. El programa está dividido en cuatro grandes bloques que se repiten en cada página web visitada. 36
3.2. Estructura de funcionamiento del bloque principal: *Descarga página web*. 37
3.3. Estructura de funcionamiento del segundo bloque principal. En él se analiza el código *HTML* descargado del bloque anterior, modificándolo y extrayendo información del mismo. 41
3.4. Estructura de funcionamiento del tercer bloque principal. El bloque *filterhtml* detecta cada línea de microdata encontrada en el archivo *htmlprocessed.html* llamando a *filtermicrodata* para su extracción. 44
3.5. Estructura de funcionamiento del cuarto bloque. En él se lleva a cabo el análisis de la microdata encontrada y extraída en la página web descargada, extrayendo dicha información junto a la imagen a la que ésta hace referencia. 46
3.6. Ejemplo de dos bloques con microdata. Puede observarse cómo el nodo *nnoderaiz* se encarga de moverse de bloque en bloque, mientras que *nnode* se mueve por el interior del bloque y el elemento *elehtml* analiza y extrae la información presente en las etiquetas. 47
3.7. Menú principal del programa donde se enumeran las distintas opciones de ejecución. 52
3.8. Ejemplo de funcionamiento. Las webs marcadas en verde son las que se han analizado según el porcentaje seleccionado, las marcadas en naranja son las que se han seleccionado para realizar una nueva iteración. 54
3.9. Ejemplo de listado de *URLs* que se desean analizar. Debe escribirse una sola *URL* por línea. 56

4.1. Ejemplo de los resultados obtenidos tras una búsqueda con el programa. Cada imagen descargada dispone de un archivo de texto con la información extraída de la microdata que la describía. 64
4.2. Gráfica de tiempo respecto al número de iteraciones (páginas de resultados) realizado. En la gráfica se representan, además, las barras de error del conjunto. 65
4.3. Imagen obtenida a partir de la búsqueda de la palabra *house* en el banco de imágenes *freefoto*. 66
4.4. Gráfica de tiempo respecto al número de iteraciones (páginas de resultados) realizado en *dreamstime*. En la gráfica se representan, además, las barras de error del conjunto. 69
4.5. Imagen obtenida a partir de la búsqueda de la palabra *cat* en el banco de imágenes *dramstime*. 70
4.6. Gráfica de tiempo respecto al número de iteraciones (páginas de resultados) realizado en *istockphoto*. En la gráfica se representan, además, las barras de error del conjunto. 73
4.7. Imagen obtenida a partir de la búsqueda de la palabra *cat* en el banco de imágenes *istockphoto*. 73

4.8. Imagen obtenida en el dominio *alanpeto.com*, Chin King, 2007. En ella se observa la estatua gigante de Buda en la provincia de Sichuan, China. La imagen dispone tanto de microdata como de datos Exif. 75
4.9. Gráfica del número de páginas analizadas en cada uno de los 100 listados del dominio. 78
4.10. Gráfica del tiempo que se ha tardado en analizar cada uno de los 100 listados del dominio junto con el número de páginas recorridas. En general puede observarse una relación directa entre el número de páginas y el tiempo realizado. 79

A.1. Menú principal del programa. En él se enumeran las distintas opciones de ejecución. Para ejecutar una opción introduzca el número de la opción seleccionada y pulse *intro*. 89
A.2. Ejemplo de listado de *URLs* que se desea analizar. Debe escribirse una sola *URL* por línea. 91
A.3. Ejemplo de archivo de resultados tras realizar una búsqueda usando la opción *Navegación por listado de urls* 92
A.4. Portada del periódico *theguardian.com* en su versión inglesa. 95
A.5. Resultados del ejemplo extraídos de la página *http://www.theguardian.com/uk*. . . . 96

Índice de tablas

2.1. Comparativa entre los distintos metaformatos existentes en la web. 24

4.1. Resultados obtenidos al realizar la búsqueda de la palabra *house* en el banco de imágenes *freefoto*. 65

4.2. Resultados obtenidos al realizar la búsqueda de la palabra *cat* en el banco de imágenes *dreamstime*. 69

4.3. Resultados obtenidos al realizar la búsqueda de la palabra *cat* en el banco de imágenes *istockphoto*. 72

Capítulo 1

Introducción

La necesidad de comunicarse es algo intrínseco en el ser humano. Las artes, el lenguaje, y una gran parte de todas las tecnologías e innovaciones que hemos desarrollado a lo largo de nuestra historia, han tenido como uno de sus principales fines el intercambio de información.

En los últimos siglos, el ser humano ha tratado a menudo de imaginar cómo sería su futuro. Ya en el siglo *XIX* algunos escritores, pioneros del género de la ciencia ficción, como Julio Verne, imaginaban que seríamos capaces de llegar a la Luna, alcanzar los polos y explorar el fondo de nuestros profundos océanos. Sin embargo, las fantasías más comunes siempre acababan con avances que cambiaban el día a día de las personas. Desde coches voladores hasta distintas maquinarias para realizar tareas cotidianas, música portátil y, sobre todo, multitud de sistemas de comunicación que permitieran hablar a distancia.

Curiosamente, muy pocos imaginaron algo como Internet. La creación de Internet ha supuesto una revolución en nuestra forma de comunicarnos de la que aún es demasiado pronto para prever sus consecuencias. La posibilidad de tener un gran volumen de información al alcance de la mano está cambiando nuestra forma de aprender. La comunicación instantánea con otras personas, a través de distintos dispositivos, cambia a su vez la forma de comunicarnos e intercambiar información. Cada vez volcamos más información de nosotros mismos en la red: datos, gustos, fechas, fotos, así como documentación tanto profesional como personal.

La información está creciendo de forma exponencial e imparable. Los motores de búsqueda, nuestras principales puertas a la información presente en Internet, tienen cada vez más dificultades para darnos la información que precisamos. Y es que la *World Wide Web*, ideada originalmente para el intercambio de documentos científicos, ha sobrepasado su finalidad original, llegando a contener una parte de nosotros mismos. Los problemas son varios. Por una parte, el gran volumen de información a indexar, así como la información duplicada, inexacta, etc. Por otra, uno de los principales escollos a superar en campos como el de la inteligencia artificial: la comprensión por las máquinas del lenguaje

natural humano.

Estos problemas requieren un cambio en la forma en la que organizamos la información. Es necesario crear nuevos conceptos más allá del tradicional de la web consistente en documentos conectados mediante hiperenlaces. Actualmente existen numerosos estudios y propuestas que tratan de reorganizar la información y ofrecer un contexto comprensible por las máquinas. Uno de los campos más importantes en este sentido es la *Web Semántica*.

La *Web Semántica*, apodada por muchos como *Web 3.0*, es uno de los principales pilares del concepto de *Big data*, entendido éste como una única red que interconecte todo el conocimiento humano, creando una consciencia común y global a partir de la interacción de cada uno de nosotros con el resto del sistema. En ese sentido, la web semántica busca la transición de la web tradicional, o de documentos, a una web en la que cada dato esté conectado y organizado a partir de conceptos más reales, propios del lenguaje y pensamiento natural humano, como objetos, personas, entidades, o incluso conceptos abstractos tales como sentimientos e ideas.

Este cambio en la forma de almacenar, organizar y presentar la información tiene como objetivo la creación de un contexto comprensible por una máquina que con la web tradicional apenas existía. Mientras que en la web tradicional una máquina es capaz de detectar la palabra *coche* en un documento, no es capaz de entender su significado y la relación de dicha palabra con el resto de información del documento. La *Web Semántica* pretende solucionar este problema mediante el uso de distintas tecnologías y vocabularios que permitan a la máquina entender que la palabra *coche* representa un objeto, que éste pertenecerá a una persona o empresa determinada, etc. En definitiva, la web semántica pretende que la información presente en Internet sea más sencilla, comprensible y accesible para las máquinas, lo que influirá directamente en la cantidad de información que éstas son capaces de procesar y manejar, ofreciéndonos, en ocasiones antes de que se lo solicitemos, más y mejor información.

A lo largo de este documento se expondrán las principales tecnologías, lenguajes y vocabularios en los que se apoya la *Web Semántica* actual. A su vez, se explicará el desarrollo del programa realizado en el proyecto, cuyo objetivo será la extracción de imágenes descritas mediante metadatos del tipo *microdata* y *Exif* para que éstos puedan ser utilizados en sistemas robóticos o agentes inteligentes.

1.1. Motivación del proyecto

La motivación del presente proyecto es la creación de un programa que permita la identificación, extracción y obtención de imágenes en la web que hayan sido descritas mediante microdata. Se pretende usar dichas imágenes con información semántica como futuro posible método alternativo para la carga de información en la base de datos de un sistema de imaginación para robots.

Este sistema, desarrollado por Víctores et al. [1], pretende ser capaz de generar modelos de objetos antes de su observación o percepción física, de forma que un robot sea capaz de tener una *imagen mental* o *concepción previa* de un determinado elemento. El sistema presenta un novedoso algoritmo de inferencia que permite, mediante el uso de *hiperplanos*, la fusión, representación y extensión del significado de determinados conceptos a partir de ciertas palabras clave. Para probar las posibilidades del sistema, éste se ha integrado en un robot cuyo objetivo final será dibujar las formas que se describan mediante órdenes de voz, basándose en el conocimiento previo existente que ha adquirido a la entrada del sistema.

Como puede observarse en la figura 1.1, la arquitectura del sistema se divide en tres partes fundamentales: *Grounding, Imagination* and *Drawing*.

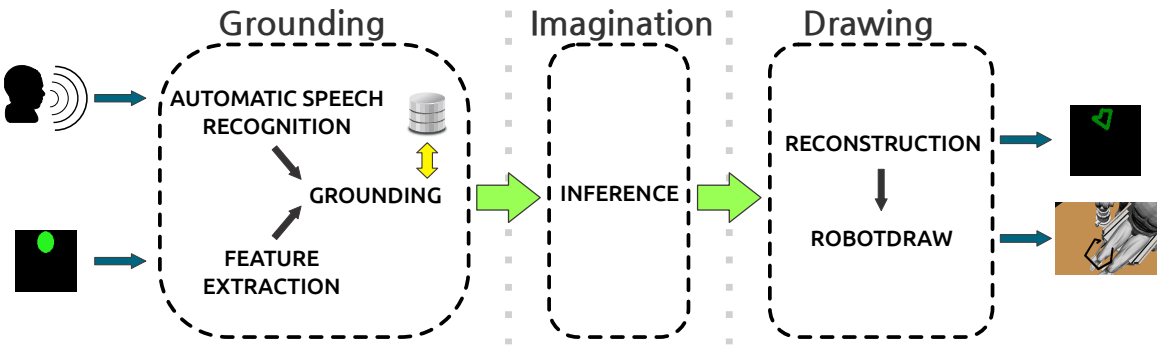

Figura 1.1: Arquitectura del sistema a alimentar con datos semánticos. Figura extraída del artículo *Towards Robot Imagination Throught Object Feature Inference*, 2013.

La primera parte o *Grounding*, se encarga de organizar, procesar y ordenar nueva información con el objetivo de alimentar la base de datos del sistema. Para ello, el sistema actualmente obtiene información a partir de dos elementos: *Automatic Speech Recognition* o reconocimiento automático de voz, que le dice al sistema palabras clave y qué se desea dibujar, y *Feature Extraction*, cuyo contenido en forma de imágenes se obtiene a partir de los datos reales de un sensor.

Con los datos obtenidos y almacenados durante el proceso de *Grounding*, la parte de inferencia denominada como *Imagination* generalizará el significado de las palabras de entrada del sistema, infiriendo, a partir de éstas y su conocimiento previo en la base de datos, una solución. Esta solución será reinterpretada y traducida a coordenadas en la última sección (*Drawing*) siendo finalmente dibujada.

El sistema, por tanto, necesita un conocimiento previo para poder realizar nuevas inferencias. Cuanto más precisa y detallada sea la información ya existente en su base de datos, mejores serán las soluciones aportadas por éste. Por ello, se entrena al sistema mediante el uso de imágenes con figuras geométricas sencillas y palabras clave asociadas a éstas (figura 1.2).

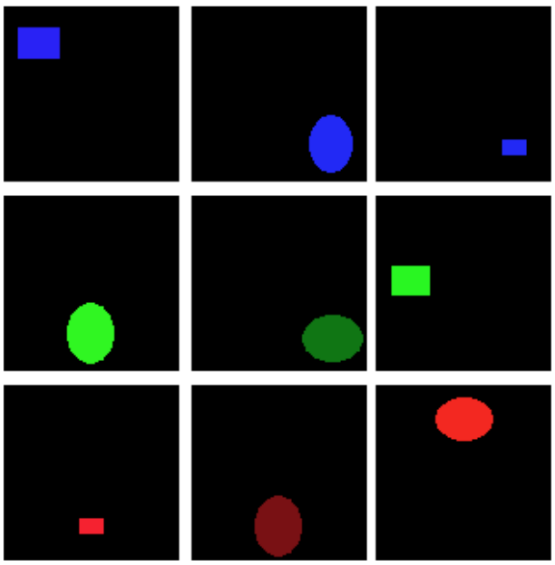

Figura 1.2: Ejemplo de imágenes de entrenamiento extraído del artículo *Towards Robot Imagination Throught Object Feature Inference*, 2013. Cada imagen tiene asociada información mediante palabras clave del tipo: rectángulo-azul-esquina-superior-izquierda.

El programa de extracción de imágenes con microdata que se explicará en éste documento pretende proporcionar una alternativa automatizada (frente al sistema de palabras clave introducidos manualmente) y real (frente al entrenamiento mediante imágenes sintéticas) para el sistema mencionado de imaginación en robots. La hipótesis básica sobre la que se trabaja es que los elementos de la nueva semántica, orientada a la información, contendrán los datos más fiables de la web actual.

1.2. Objetivos específicos

Se pretende obtener imágenes, con información clave asociada a éstas, de forma automática, a través de la exploración y análisis de las páginas web presentes en Internet. Para lograr este objetivo, el programa será la combinación de un programa araña web o *crawler* y un analizador o programa *parser* de *HTML* con microdata. El presente proyecto tendrá los siguientes objetivos específicos:

- **Desarrollo de un programa extractor de imágenes con *microdata*:** El principal objetivo del proyecto es la creación de un programa tipo *crawler*, o *araña web*, que sea capaz de analizar la web en busca de imágenes descritas mediante metadatos del tipo *microdata*. Se pretende que la extracción de dichas imágenes, junto a la información aportada por la *mi-*

crodata, pueda ser una alternativa real para alimentar bases de datos de distintos sistemas robóticos o agentes inteligentes, como el desarrollado por Victores et al. [1], cuya descripción puede encontrarse en el apartado anterior.

- **Estudio del estado de la Web Semántica:** Se pretende obtener una visión general del estado actual de la web semántica (apartado 2.1), así como de las distintas tecnologías que componen ésta, y su futuro próximo.

- **Extracción de datos *Exif*:** Se pretende extraer los datos *Exif* (apartado 3.2.1) presentes en las imágenes descargadas, con el objetivo de que dichos datos complementen la información aportada por la microdata. Además, se pretende realizar una primera aproximación a la interpretación del contenido, u objetivo de la fotografía realizada, a partir de los datos *Exif* obtenidos por un programa como el presente.

1.3. Estructura del documento

A continuación se detalla la estructura del documento con el objetivo de facilitar su lectura:

- En el capítulo 1 se realiza una introducción del proyecto.

- En el capítulo 2, el *Estado del arte*, se realiza un repaso a la situación actual de la web semántica, desde sus inicios junto con la web original, a los últimos avances y vocabularios implementados. A continuación se explica con mayor detalle el sistema de metadatos denominado como *microdata*. Por último, se realiza una descripción de los programas especializados en la exploración de la web, comúnmente denominados como *arañas web* o *crawlers*.

- En el capítulo 3 se realiza una descripción del programa *Web Spider Microdata* desarrollado, desde sus objetivos y requisitos, lenguaje y librerías utilizadas, así como su funcionamiento básico, hasta una descripción más pormenorizada de los aspectos clave de cada uno de los bloques que definen su arquitectura. Por último, se presentan los distintos ensayos realizados para comprobar el comportamiento del programa ante distintos casos encontrados en la web, exponiendo y justificando a continuación sus resultados.

- En el capítulo 4 se exponen las conclusiones del proyecto, comentando la situación actual de la web semántica así como sus aplicaciones y evolución futura, y se exponen los posibles aspectos a desarrollar a partir del proyecto presente.

Capítulo 2

Estado del arte

En este capítulo se expondrá la evolución de la web semántica, ligada a la creación de la propia web, así como su estado actual y posible evolución futura, explicando, además, los distintos conjuntos de tecnologías que la forman. A continuación se explicará con mayor detalle el vocabulario de metadatos conocidos como *microdata*, objetivo de extracción del programa realizado, explicando su funcionamiento y posibles aplicaciones. Por último se explicará qué es un programa tipo *araña web* o *crawler*, exponiendo su funcionamiento básico así como sus principales problemas y aplicaciones actuales.

2.1. Web semántica

La web semántica, muchas veces denominada como *Web 3.0*, es un conjunto de herramientas y tecnologías que, sobre la web actual, trata de abordar uno de los principales problemas de la información publicada en Internet desde su nacimiento: la pérdida de datos fácilmente interpretables por máquinas y personas, así como su pérdida de contexto e interrelaciones con otros datos.

Una definición más formal de la web semántica puede ser la realizada por Hendler, Berners-Lee y Miller [2]:

> «La web semántica es una extensión de la actual web en la que a la información disponible se le otorga un significado bien definido que permita a los ordenadores y a las personas trabajar en cooperación. Está basada en la idea de proporcionar en la web datos definidos y enlazados, permitiendo que aplicaciones heterogéneas localicen, integren, razonen y reutilicen la información presente en la web.»

Así pues, se considera a la web semántica como una extensión de la web actual y por lo tanto sus aplicaciones y herramientas deben coexistir con ella. Esta web debe proporcionar información

marcada con datos que aporten un significado bien definido y, por tanto, capaz de ser compartido y enlazado de forma que sea posible su reutilización por diferentes entidades.

En último lugar, es importante destacar que la web semántica tiene también como objetivo que los sistemas informáticos puedan ser capaces de manipular información, llegando incluso a reelaborarla, con objetivos concretos según los problemas planteados.

En definitiva, la web semántica pretende interconectar a las personas y máquinas con los datos de una forma más eficiente.

2.1.1. Evolución de la web, el paso de la web de documentos a la web social, la transición hacia la web de datos

Desde el nacimiento de Internet a finales de los años sesenta, éste estuvo centrado en el intercambio de información, sobre todo mediante el intercambio de correos electrónicos a través de listas de correo. Sin embargo, hasta principios de los años noventa no surgió la que sería su principal medio de intercambio de información. La web se originó mediante la unión de dos herramientas ya existentes: el hipertexto e Internet.

El hipertexto, concebido por Vannevar Bush durante la segunda guerra mundial, se basa en la idea de una biblioteca de conocimiento colectivo con enlaces activos entre documentos [3]. Fue Ted Nelson quien posteriormente bautizó a este concepto como *hipertexto* [4]. Este primer intento inspiraría varios sistemas como el *NLS* de Douglas Engelbart o el sistema *Hypercard* de *Apple* [5].

Uniendo estas dos herramientas Tim Berners-Lee, un contratista independiente del *CERN*, empezó en 1980 a desarrollar *Enquire* [6], una base de datos personal de gente y modelos de software en la que (basándose en la idea del hipertexto) cada nueva página de información debía de estar conectada a una página ya existente.

En 1984 Berners-Lee empezó a considerar los problemas de intercambio de información que se daban en la ciencia del momento. Los investigadores necesitaban compartir datos pero sus computadoras y software de presentación eran diferentes y en muchas ocasiones incompatibles. Presentó por ello una propuesta en 1989 para "Una gran base de datos de hipertexto con enlaces tipados" que tras varios nombres acabaría por denominarse como *World Wide Web*.

En agosto de 1991 se creo la primera página web del mundo [7]. Surgió así una nueva herramienta en Internet en la que el usuario podía acceder a una serie de documentos en forma de hipertexto. Berners-Lee desarrolló para ello todas las herramientas que se requerían para trabajar en la web. El protocolo de transferencia de hipertexto o *http*, el primer navegador web que permitía tanto la lectura como edición de páginas web, el primer servidor de aplicaciones *http* y el lenguaje de marcado de hipertexto o *HTML*.

En tan solo cinco años, la popularidad de la web rebasó el ámbito científico y académico dándose el auge del comercio electrónico con el nacimiento de empresas como *Amazon, Ebay o Yahoo*. Se

originó también la primera guerra de navegadores entre *Internet Explorer y Netscape*. Es en este momento cuando Tim Berners-Lee crea el consorcio de la *World Wide Web*, conocido comúnmente como *W3C*, que de forma independiente se encargará de la estandarización del lenguaje *HTML*. Será este consorcio quién cree en 1996 el lenguaje *XML* [8] para facilitar el intercambio de información estructurada debido al auge y necesidades del comercio electrónico.

La aparición del comercio electrónico originó una burbuja especulativa en la que se crearon una gran cantidad de las denominadas empresas *puntocom*. Este movimiento alcanzó su auge en el año 2000, a partir del cual el mercado de las empresas *puntocom* sufrió una pérdida de apoyo que terminó en el cierre y reestructuración de las empresas de Internet.

En esta situación cuando empezó a surgir lo que hoy denominamos *Web 2.0*. Una web que evolucionaba junto con el resto de tecnología de la información y permitía que la web se centrará en la comunicación entre personas en vez de en el intercambio de documentos. Así la web deja de considerarse por los usuarios como un sistema de solo lectura y pasa a ser un sistema de lectura/escritura donde éstos participan en el contenido. Surgen aplicaciones web como lectores de *email*, editores de texto e imágenes y otras, que son cada vez más elaboradas. Esta nueva visión de la web origina la creación de elementos como los blogs personales, las redes sociales y proyectos de inteligencia colectiva en la que los usuarios participan en la generación de nuevo conocimiento. Un ejemplo de esto último es la enciclopedia online *Wikipedia*.

La *Web 2.0* comienza a ser importante por sus datos, por lo que surgen nuevas librerías de programación que fomentan la creación de otras *interfaces* o aplicaciones realizadas por terceros. Estas aplicaciones, creadas a partir de los datos y librerías de otras aplicaciones, se agrupan bajo el término de *mashups* siendo un ecosistema de aplicaciones interconectadas. La revolución de los dispositivos móviles junto a estas nuevas librerías de programación y la necesidad de adaptar estos datos a *interfaces* y formatos nuevos incrementarán la importancia de los datos en la web frente a su formato de presentación. A partir del año 2009 la *Web 2.0* se volcará con las redes sociales y primará la inmediatez en la información. Redes sociales como *Facebook, Twitter o LinkedIn* cambian la forma en la que el usuario accede, comparte y recibe contenido a través de Internet.

En la actualidad, en lo que se puede denominar madurez de la *Web 2.0*, la visión de la web ha cambiado de una red de intercambio de información de documentos a un progresivo intercambio y reutilización de datos individuales que se actualizan constantemente.

Como puede observarse, se está realizando una progresiva evolución de la web sintáctica (donde la unidad de información es el documento) a una web semántica o de datos (donde la unidad pasa a ser el propio dato).

Sin embargo, la idea de la web semántica, o de datos, no es nueva. La denominación de la web semántica como *Web 3.0*, aun siendo un termino común para nombrarla, no está totalmente aceptada por los expertos. El propio Tim Berners-Lee defiende que la web es un invento único que va

evolucionando de forma continuada y que por lo tanto carece de sentido hablar de versiones dentro de ella. Ya desde la primera conferencia de la *W3C* en 1994, Berners-Lee mencionaba la necesidad de añadir el término *semántica* a la web [9].

2.1.2. Estructura de la web semántica

La web semántica se apoya en el paso de la web de documentos a la web de datos. La estructura de la web de documentos o *Web 1.0* (figura 2.1) se caracteriza por ser una serie de documentos de hipertexto relacionados entre sí.

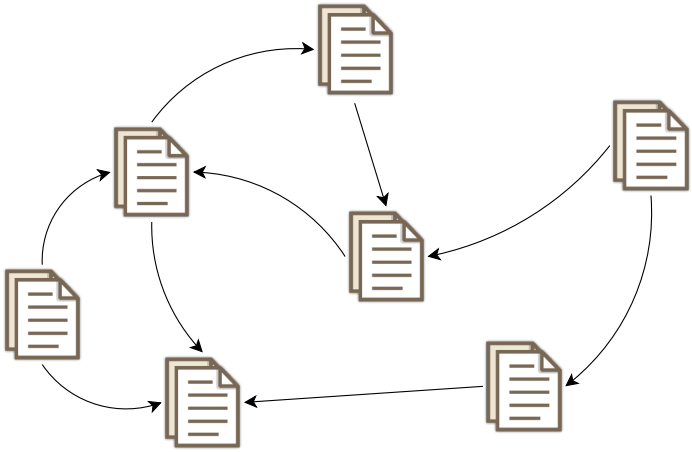

Figura 2.1: Estructura de la web de documentos: la web se entiende como una red de documentos relacionados siendo éste su unidad más pequeña.

Esta estructura, donde la unidad más pequeña de información es el documento, se ha presentado insuficiente con la evolución de la web . Existe una pérdida de contexto evidente que dificulta tanto a personas como a máquinas el acceso, reutilización y manipulación de datos para múltiples necesidades.

Por ello, con el desarrollo de la *Web 2.0*, los datos han ido cobrando cada vez más importancia. El valor añadido de empresas como *LinkedIn, Facebook y Google* radica en los datos que manejan y que pueden ofrecer al cliente, sea usuario o empresa, con múltiples aplicaciones. Un ejemplo de esta necesidad de manipular datos en la web la encontramos en *Facebook* y su modelo de negocio, que se basa en hacer llegar la publicidad de las empresas directamente a su público objetivo, basándose en los gustos, edad, y otros datos similares de cada usuario. En la figura 2.2 podemos observar una estructura típica de información en la *Web 2.0*.

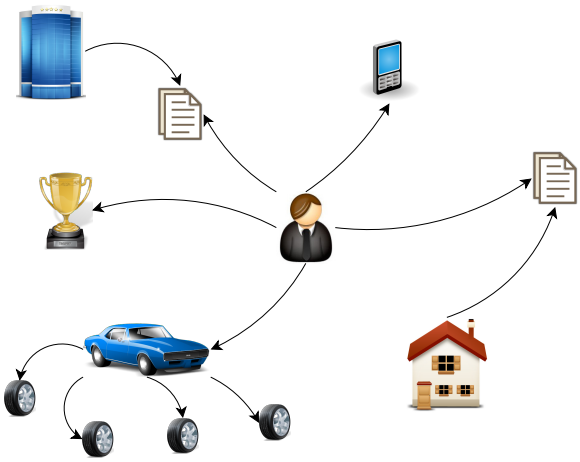

Figura 2.2: Ejemplo de la web de datos. Cada objeto representa un ente real o concepto abstracto relacionados entre sí.

Pero el paso de una web de documentos a una web de datos no es suficiente. Los datos crudos no se comparten con el resto de la red, por lo que cada organismo u empresa genera y mantiene su propia base de datos, lenguaje e infraestructura. Esto origina una gran cantidad de datos duplicados que, en la mayoría de las ocasiones, no están relacionados entre sí (figura 2.3).

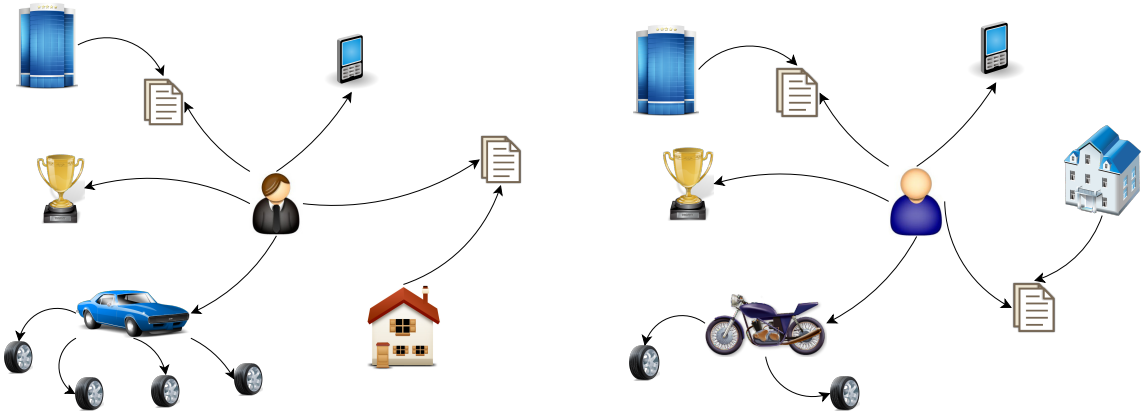

Figura 2.3: Ejemplo de dos conjuntos de datos no relacionados entre sí. Las dos personas representadas trabajan para la misma empresa y ambas tienen el mismo modelo de teléfono móvil. Sin embargo ambos datos están duplicados y los datos de los dos conjuntos no se interrelacionan.

De la necesidad de aportar una estructura, organización y contexto a datos crudos surge la web semántica. La web semántica pretende interrelacionar todos esos datos, evitando duplicidades y facilitando la reutilización de datos ya existentes para crear una única red de redes de datos, a partir de los cuales pueda elaborarse información capaz de usarse en diversos ámbitos (figura 2.4).

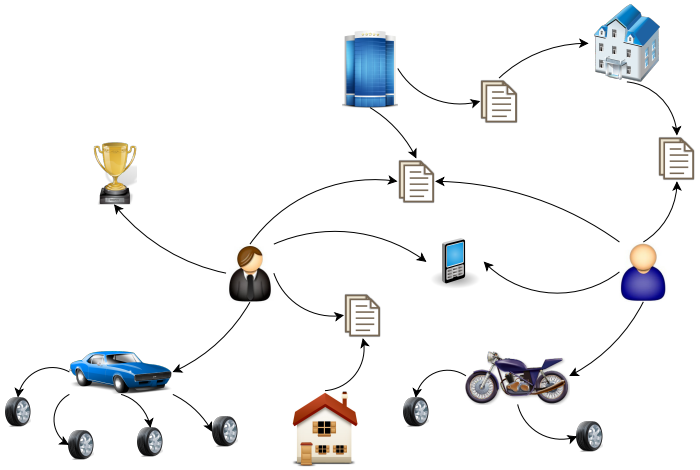

Figura 2.4: Ejemplo de web de datos usando la web semántica.

En la figura 2.4 los datos comunes, como que ambas personas trabajan para la misma empresa o poseen el mismo modelo de teléfono, no se presentan duplicados. Las interrelaciones permiten además descubrir más información, como que la casa del sujeto de la derecha está arrendada por la empresa en la que ambos trabajan.

En el año 2000 se propuso un modelo de capas que esquematizara el desarrollo de la web semántica (figura 2.5). Se bautizó como *Semantic Web Layer cake* o Tarta Semántica [10].

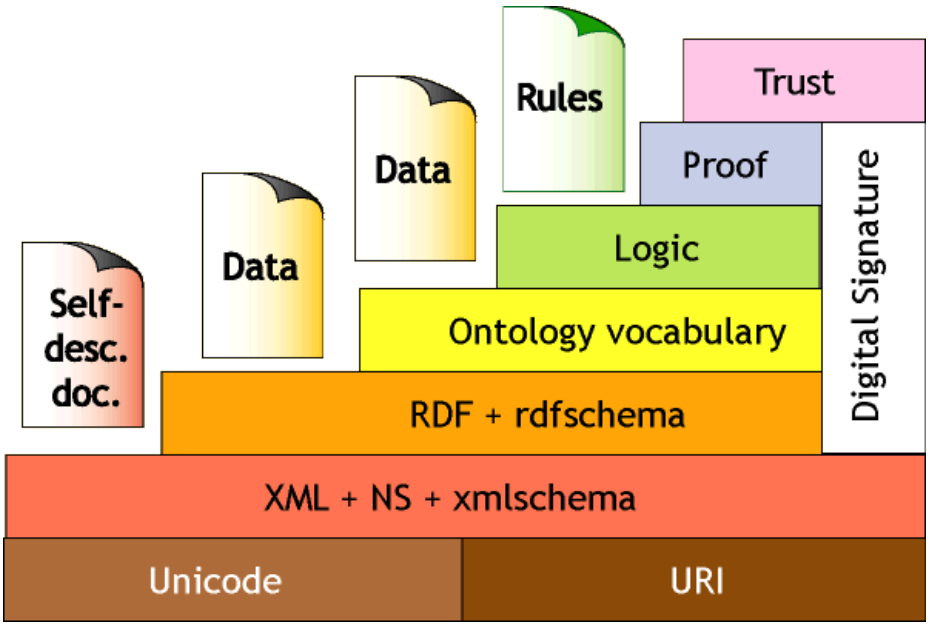

Figura 2.5: Tarta Semántica propuesta por Tim Berners-Lee, 2000

Este modelo se basa en la premisa de que cada capa corresponde a una tecnología que se apoya en la tecnología de la capa anterior. A partir de esta premisa se explicarán los componentes de cada capa así como su relación con el resto del modelo:

- **Unicode y URI (Codificación y Localización):** La primera capa agrupa las dos tecnologías de infraestructura más básicas. *Unicode* es un estándar de codificación en múltiples idiomas. *URI*, al igual que en la web actual, será la tecnología usada para identificar recursos. Una *URI* puede representar desde páginas webs, imágenes, ficheros, etc., hasta entidades reales o incluso ideas abstractas. Cuando la *URI* tiene una dirección accesible en la web, se denomina como *URL*.

- **XML (Sintaxis):** El lenguaje *XML* se define como una de las principales herramientas de sintaxis a la hora de desarrollar el resto de las capas de la Tarta Semántica. Creado con el propósito de almacenar datos estructurados de forma legible, *XML* es un lenguaje ampliamente usado en la gestión de bases de datos.

  Cabe destacar el uso de los espacios de nombres (*XML namespaces*) en la web semántica. Éstos permiten indicar que en un mismo documento van a utilizarse diferentes vocabularios *XML*, lo que evita conflictos en definiciones similares de vocabularios diferentes. De esta forma, cada vocabulario será un *namespace*, donde cada elemento tendrá un nombre particular asegurándose que dicho elemento tiene una única definición y sigue unas reglas específicas dentro del *namespace* declarado.

  En el momento de la publicación de este esquema, *W3C* apostaba por la versión *XML* de *HTML* conocida como *XHTML*.

- **RDF y RDF Schema (Descripción y estructura):** *RDF* y *RDF Schema* crean la infraestructura que permitirá la descripción de los recursos. Estos lenguajes se basan en la creación de tripletas del tipo *sujeto-predicado-objeto* que permite el uso de grafos para ampliar y relacionar distintas tripletas.

- **Ontology vocabulary:** Según el artículo *Knowledge engineering: principles and methods* [11]:

  > «Una ontología es una especificación formal y explícita de una conceptualización compartida.»

  El objetivo de las ontologías en la web semántica es la de describir objetos mediante la definición de clases, propiedades, relaciones y axiomas. Un ejemplo sencillo sería una ontología para un catálogo de elementos multimedia. En ella, la película *Metrópolis, Fritz Lang (1927)*, como puede observarse en la figura 2.6, sería un objeto tipo *película* descrita por propiedades como *título, descripción, año de publicación, idioma etc.* A su vez estaría relacionada con otros objetos como *director*, en este caso *Fritz Lang*, que dispondría a su vez de propiedades como *nombre, año y lugar de nacimiento, año de defunción, películas que ha hecho, etc.*, así como actores, medios de publicación y otros elementos relacionados.

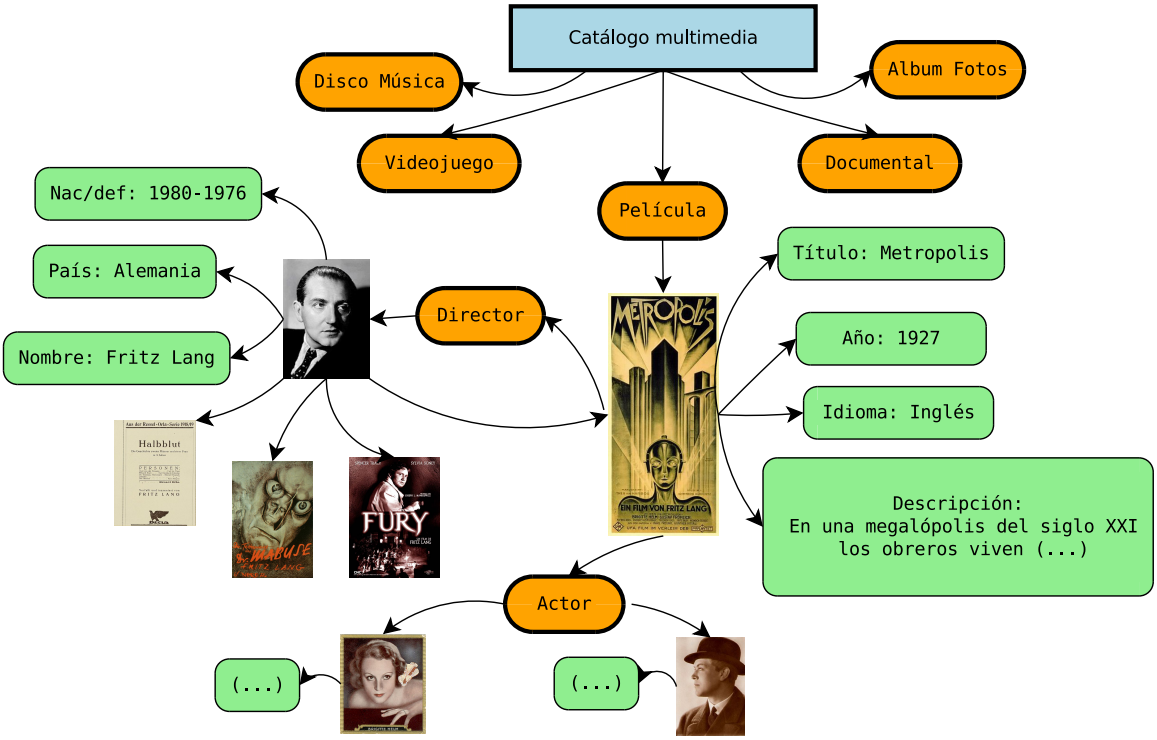

Figura 2.6: Ejemplo de ontología para un catálogo multimedia.

En definitiva, las ontologías definen, relacionan y formalizan un vocabulario común para un área de conocimiento dada.

- **Logic, proof:** En estas dos capas superiores se identifican los aspectos de inferencia que se deben desarrollar para crear nuevo conocimiento a partir de los datos existentes. Un ejemplo de ello sería el conjunto de reglas para usar conjuntamente dos ontologías distintas, o cómo relacionar y usar elementos de una u otra.

- **Trust (Criptografía y firma digital):** La capa superior surge de la preocupación acerca de si las declaraciones realizadas en la web semántica son fiables o no. Para ello se debe desarrollar una infraestructura que se base en la tecnología de firmas digitales, presente en todas las capas de la Tarta Semántica.

Esta definición de la estructura de la web semántica tuvo modificaciones importantes. Al principio se pretendía que cada capa de un nivel se basase en el nivel de la capa inmediatamente anterior. Por ejemplo que el lenguaje *RDF* se basase en las tecnologías *XML*. Sin embargo, los desarrollos posteriores demostraron que, si bien esta regla era altamente recomendable, no era posible en todas las ocasiones. Además se observó una carencia importante en la organización propuesta al no existir una infraestructura para realizar consultas. Por ello, en el año 2006 se publicó una nueva versión de la Tarta Semántica (figura 2.7) [12].

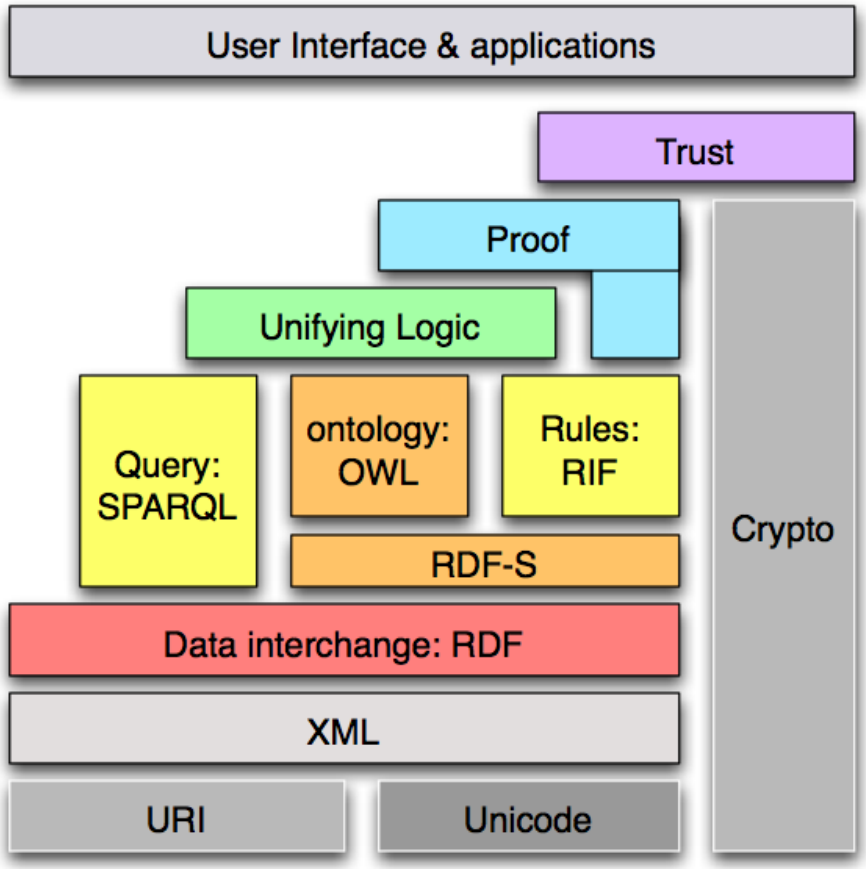

Figura 2.7: Nueva Tarta Semántica propuesta por el W3C, 2006

Las principales novedades son:

- **Lenguaje OWL:** La capa de ontologías pasará a identificarse con el lenguaje *OWL*, que es una evolución del lenguaje *RDF* cuyo objetivo es la definición de ontologías.
- **Lenguaje de consulta SPARQL:** Será el lenguaje de consultas usado en la web semántica, siguiendo un funcionamiento similar al de *SQL* en las bases de datos relacionales.
- **Reglas RIF:** Debido a la existencia de distintos sistemas dentro de la web semántica, surge la necesidad de desarrollar *Rule Interchange Formart o RIF*. Una infraestructura que permite definir reglas de un modo interoperable entre distintos sistemas.

Sin embargo, esta Tarta Semántica no es la definitiva. Junto a las dificultades técnicas para asegurarse que cada capa superior esté basada en la capa inferior, muchas de sus tecnologías aun se están desarrollando y surgen nuevas propuestas como *RDFa o Turtle* que presentan dificultades para encajar en la estructura. Además aun no se ha desarrollado de forma completa, una infraestructura

de criptografía y firma digital, una capa de confianza, por lo que uno de los principales problemas a los que se enfrenta la web semántica, la fiabilidad y confianza de los datos, aún necesita desarrollo.

Sin duda alguna, en un futuro surgirán nuevas versiones de la Tarta Semántica. Por ello ésta no debe seguirse de forma literal, sino que debe utilizarse como un mapa conceptual de sus tecnologías.

2.1.3. Aplicaciones de la web semántica

Las aplicaciones de la web semántica son numerosas y pueden ser aplicadas a un amplio rango de información en la web. Como ya se mencionó, los principales principios de la web semántica nacieron prácticamente al mismo tiempo que la web original, por lo que muchas de sus aplicaciones están integradas en ésta desde el nacimiento de la web. Otras, sin embargo, pretenden, usando *RDF* y los metaformatos, realizar nuevas mejoras en la información que recibe el usuario y lo que es capaz de entender una máquina. A continuación se expondrán algunos ejemplos de cómo la web semántica se está expandiendo cada vez más como una capa nueva en la web tradicional.

Informes financieros

El sector financiero es un sector fuertemente informatizado en el que la mayoría de las operaciones se realizan por parte de máquinas. Existe un gran interés por aplicar tecnologías de la web semántica que permitan transformar documentos financieros *XBRL* en formatos semánticos como *RDF*. Un ejemplo llamativo es el proyecto *Semantic XBRL* del grupo Griho de la Universidad de Lleida [13], donde se han transformado los documentos financieros del sistema estadounidense *EDGAR* a *RDF*, obteniéndose más de 46 millones de declaraciones *RDF*.

Semántica en la documentación médica

Las tecnologías de la web semántica tienen grandes oportunidades en la documentación médica. Un sector donde una clasificación eficaz de la información y el acceso a ésta es fundamental, debido a la enorme cantidad de datos a tratar. Si un médico dispone de un sistema adecuado, será capaz de diagnosticar de forma más rápida y eficaz el problema de un paciente. En este sentido, la web semántica puede aportar grandes ventajas, como la integración entre distintos sistemas de información médicos, así como realizar nuevas inferencias que permita mejorar el diagnóstico de ciertas enfermedades a partir de diversos síntomas.

Un ejemplo de ello puede ser la integración y representación de los sistemas de visualización de datos. En el artículo *Semantic enhacement of electronic health records for decision support and interactive information visualization* [14], se realiza una propuesta denominada como *V.A.F Framework*. Ésta pretende la integración de los estándares de documentación médica como *EHRs* y *CDA*, usando tecnologías de la web semántica como *OWL* para la creación de ontologías accesibles a través de la web, y *SPARQL* para la recuperación eficaz de dicha información.

Otros de los muchos proyectos orientados a la web semántica en el campo médico es el proyecto *The Cancer Genome Atlas* [15], cuyo objetivo es catalogar las mutaciones genéticas responsables del cáncer. Debido a la gran cantidad de información existente, en la Universidad de Texas están desarrollando un sistema que transforma los datos a tripletas *RDF*, ofreciendo además una terminal *SPARQL Endpoint* para su consulta [16].

Raw data y mashups

El fenómeno de la web de datos, donde cada vez más instituciones, empresas y usuarios comparten datos crudos en Internet, ha originado una gran cantidad de aplicaciones que usan dichos datos de formas muy diversas y, en muchas ocasiones, de un modo totalmente distinto al propósito original para el que se crearon.

Dichas aplicaciones, encargadas de representar e interpretar estos datos crudos, son popularmente conocidas como *mashups*. Existen muchos ejemplos en la web de aplicaciones con estas características de un alto contenido semántico. Algunas de las más famosas son usadas junto con *Google maps* o *Openstreetmap*, donde los usuarios crean capas de mapas en las que se superponen datos actualizados de diversa índole, desde la concentración de accidentes de coches, a mapas colaborativos y en tiempo real de zonas afectadas por desastres naturales, como el ocurrido en Haití en el año 2010 [17].

Destacan cada vez más aplicaciones realizadas a partir de datos gubernamentales, como en qué se está gastando el dinero de los impuestos en Gran Bretaña o denunciar el estado de una calle a las autoridades para que éstas procedan a su reparación, así como los resultados de las votaciones de varios países. O propuestas como *Layar*, un navegador de realidad aumentada que permite visualizar puntos de interés cercanos al usuario, o *Waze*, un sistema de ayuda a los conductores, colaborativo, donde se puede alertar a otros conductores de accidentes, congestiones de tráfico, etc.

En definitiva, las *mashups* son aplicaciones orientadas a representar gráficamente una serie de datos que faciliten poder llegar a conclusiones, o realizar nuevos descubrimientos, de una forma más rápida y eficaz.

Linked data y bancos de información semántica

La web semántica es ya una red formada por millones de datos. Si se observa su diagrama de etiquetas en la figura 2.8, se puede percibir que destacan ciertos dominios que se comentarán a continuación:

- **DBpedia:** *DBpedia* es un proyecto que nació con el objetivo de obtener datos semánticos a partir de los contenidos de la enciclopedia colaborativa *Wikipedia*. Coordinado por la Universidad de Lepzig y la Universidad libre de Berlín, en el año 2011, contenía más de tres millones y medio de objetos y más de mil millones de sentencias *RDF*. *DBpedia* contiene distintas subclases de recursos. Desde personas a lugares, trabajos, organizaciones, etc., cada una con sus

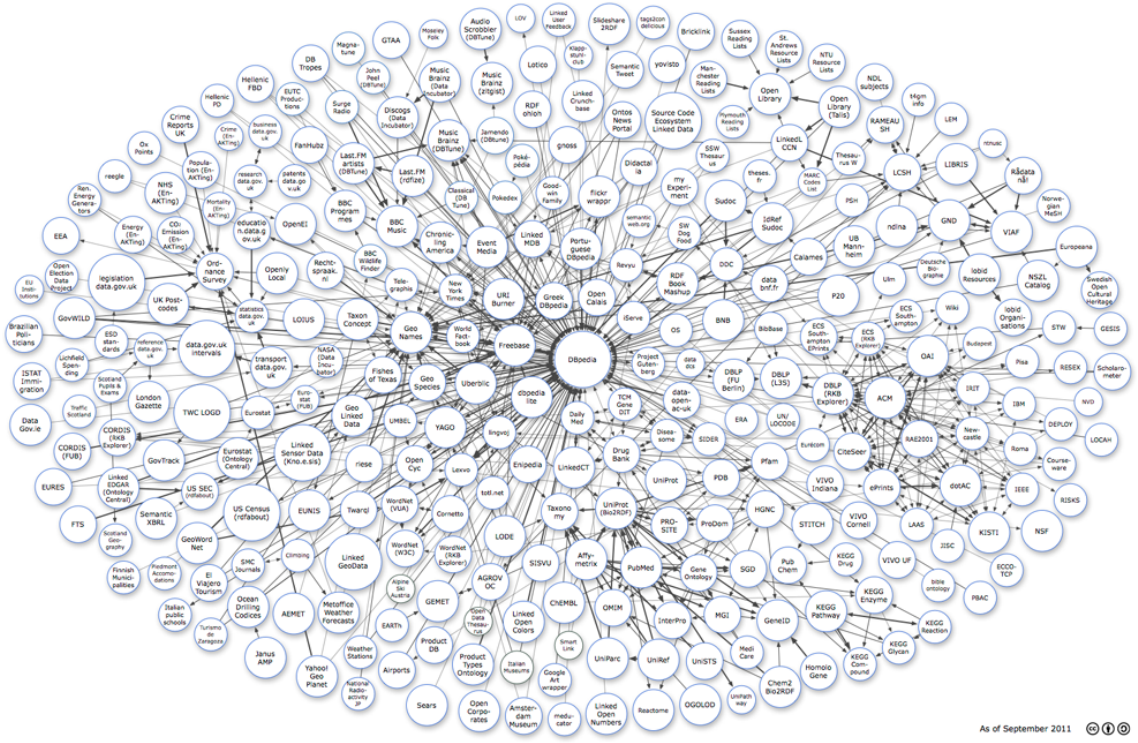

Figura 2.8: *The linked open data cloud diagram*, linkeddata.org, 2011 muestra los principales dominios de Internet que contienen información semántica y su relación entre ellos. Destaca el proyecto *DBpedia* como uno de los nodos más relacionados.

respectivas propiedades. *DBpedia* es ampliamente utilizado por otros dominios, que usan sus datos dentro de la web semántica con distintos fines.

- **Freebase:** Es uno de los principales nodos de información semántica presentes en la web. Mezcla la definición automática de nuevos objetos a partir de información de otros nodos, como *DBpedia*, con la colaboración de los usuarios, siendo un gran medio para encontrar información de forma ordenada y recopilada de una entidad determinada.

- **Europeana:** *Europeana* nació para compartir, recopilar y permitir el acceso al patrimonio cultural europeo con el objetivo de difundirlo y preservarlo. En el proyecto participan más de 1.500 instituciones culturales, como bibliotecas, museos, archivos y asociaciones culturales de toda Europa. *Europeana* no funciona como un servidor principal en el que se almacenan todos los objetos digitalizados, sino como un catalogo centralizado que relaciona los distintos repositorios de cada institución. Para ello, usa distintas tecnologías propias como *ESE (Europeana Semantic Elements* que mezclados con tecnologías como *RDF, SKOS* y *SPARQL* proporcionan la integración y coherencia de los datos recogidos de las diversas fuentes europeas.

Microformatos, RDFa y microdata

Los denominados como microformatos permiten la integración de información semántica en el propio código de las páginas web. Sus aplicaciones y expansión, debido a su relativa sencillez y eficacia de descripción, hacen que sean ampliamente usados en numerosos sitios web, desde bancos de imágenes a páginas de productos, críticas de cine y televisión, webs de recetas, etc. Su uso permite una mejor indexación por parte de los motores de búsqueda, que ofrecen de esta forma información más precisa y detallada al usuario.

A su vez, este tipo de formatos semánticos son ampliamente utilizados en las redes sociales, generando relaciones entre entidades como personas, empresas, gustos, aficiones, etc. Estas relaciones permiten a las redes sociales ofrecer mejor contenido al usuario, así como monetizar su información. En este ámbito destaca *Facebook* con su *Facebook Open Graph* [18], que permite relacionar las actividades de un usuario en otros servicios web con la red social.

2.2. Microdata y HTML5: Acercando la web semántica al diseño web

El 2 de Junio de 2011 los principales motores de búsqueda *Google, Bing* y *Yahoo* anunciaron la creación del sitio web *schema.org* [19] con el fin de promover un nuevo estándar de datos semánticos incrustados: la microdata. Microdata es un formato que pretende añadir pequeñas descripciones semánticas sobre entidades como libros, personas, lugares, empresas etc., dentro del código *HTML5*.

El estándar *HTML5* nació con el fin de hacer la web menos dependiente de lenguajes como *Java* y *Flash*, potenciando y promoviendo los elementos multimedia. Al mismo tiempo el nuevo estándar tenía un reto aún mayor: aumentar la expresividad y el contexto de la información en la web. *HTML5* contiene ya elementos semánticos como las etiquetas *ol, header, nav, article, aside, section o footer*. Sin embargo, estas etiquetas dan más detalle a los navegadores y otros programas sobre el formato de la página, y no sobre su contenido.

Anteriormente, para describir a una persona con código incrustado, es decir, sin tener que recurrir a crear páginas de tripletas *RDF*, la única opción era usar los microformatos y su etiqueta *class*. Aunque hoy en día su uso sigue estando muy extendido, su expresividad y eficacia son limitadas, por lo que surgió la propuesta del *W3C* de adaptar la estructura básica de *RDF* a un código incrustado en *XHTML*, creando *RDFa*. Si bien es una forma sencilla de describir tripletas de forma incrustada, seguía siendo una solución compleja para muchos diseñadores web que continuaron usando los microformatos para conseguir mayor información semántica y un mayor posicionamiento en los resultados de los motores de búsqueda.

2.2.1. Schema.org: El esquema adoptado por los grandes buscadores

Como ya se ha comentado, en 2011 los motores de búsqueda *Google, Bing* y *Yahoo* crean *schema.org* [19], presentando un nuevo formato semántico incrustado. Actualmente colabora también *Yandex*, el principal buscador ruso de la web. *Schema.org* se basa en intentos anteriores como *RDFa* y los microformatos para realizar la estructura de la microdata. Todas sus entidades están vinculadas a una superentidad o entidad madre denominada *Thing*. A partir de ella se describen distintos tipos de información, desde personas a productos, imágenes, páginas web, vídeos etc., así como sus respectivos subtipos.

2.2.2. Estructura de la microdata

La microdata usa atributos de *HTML5* para describir las entidades. Toda la microdata de una entidad se describe dentro de la misma etiqueta. De esta forma, la información semántica puede estar incrustada en cualquier etiqueta propia de *HTML5*, si bien lo habitual es el uso de etiquetas contenedoras como *div* o *span*. A los atributos ya existentes en *HTML5* se les incorporan cinco más orientados al uso de la microdata, que se enumeran a continuación:

- **itemscope:** Su propósito es indicar que se empieza describir una entidad descrita con microdata. Aunque formalmente es un atributo sin valor, y va directamente sucedido por *itemtype*, en la práctica muchas páginas web lo sustituyen por *itemscope=“”* o *itemscope=“itemscope”* para facilitar a los distintos programas de análisis web la lectura de la microdata, debido a que éstos habitualmente presentan problemas de incompatibilidad a la hora de leer atributos vacíos. Un ejemplo de cómo comenzaría una etiqueta contenedora con microdata sería:

```
<div itemscope itemtype="http://schema.org/Offer">...</div>
o
<div itemscope="" itemtype="http://schema.org/Offer">...</div>
o
<div itemscope="itemscope" itemtype="http://schema.org/Offer">...</div>
```

- **itemtype:** Tras indicar con *itemscope* el comienzo de un bloque de microdata, se indica el tipo de entidad que se va a describir mediante el atributo *itemtype*. En el ejemplo anterior puede verse como *itemtype* tiene como valor una *URL* de *schema.org* que lo identifica como un objeto tipo oferta. De esta forma se indica el tipo de entidad y a su vez se direcciona a la página que contiene los valores de dicha entidad.

- **itemprop:** El atributo *itemprop* es el encargado de definir las propiedades a declarar de la entidad. El valor de cada entidad será normalmente escrito entre las etiquetas, si bien en

algunas ocasiones y propiedades el valor estará en otro atributo como *content, src, href*, etc. Se pueden observar los dos casos posibles en el siguiente ejemplo:

```
<span itemprop="name">Blend-O-Matic</span>
o
<span itemprop="name" content="Blend-O-Matic"/>
```

- **itemid:** Permite definir un identificador global para la entidad descrita, por ejemplo el número *ISBN* de un libro.

- **itemref:** Permite describir dentro de la identidad propiedades que no son propias de ésta, es decir, propiedades que no están directamente relacionadas con la entidad descrita ni son un subtipo de ésta. Un ejemplo sería una entidad que describe una imagen tipo *imageobject* de temática médica que muestra un conjunto de huesos en ésta. La entidad *imageobject* no puede en principio describir datos médicos. Sin embargo, y mediante el uso de *itemref*, puede describirse dentro de ésta la entidad *bone* que aportará información semántica útil en el ámbito médico.

Si bien los dos últimos atributos, *itemid* e *itemref*, pueden usarse para ciertos casos, en la práctica lo más común es el uso de los tres atributos: *itemscope, itemtype* e *itemprop*. De esta forma, con solo tres atributos adicionales en *HTML5*, podemos describir multitud de clases distintas en una página web.

A continuación se muestra un ejemplo sencillo de persona descrita con microdata, extraído de *schema.org* [20]:

```
<div itemscope itemtype="http://schema.org/Person">
  <span itemprop="name">Jane Doe</span>
  <img src="janedoe.jpg" itemprop="image" />

  <span itemprop="jobTitle">Professor</span>
  <div itemprop="address" itemscope itemtype="http://schema.org/
      PostalAddress">
    <span itemprop="streetAddress">
      20341 Whitworth Institute
      405 N. Whitworth
    </span>
    <span itemprop="addressLocality">Seattle</span>,
    <span itemprop="addressRegion">WA</span>
    <span itemprop="postalCode">98052</span>
```

```
14      </div>
15      <span itemprop="telephone">(425) 123-4567</span>
16      <a href="mailto:jane-doe@xyz.edu" itemprop="email">
17        jane-doe@xyz.edu</a>
18
19      Jane's home page:
20      <a href="http://www.janedoe.com" itemprop="url">janedoe.com</a>
21
22      Graduate students:
23      <a href="http://www.xyz.edu/students/alicejones.html" itemprop="colleague"
            >
24        Alice Jones</a>
25      <a href="http://www.xyz.edu/students/bobsmith.html" itemprop="colleague">
26        Bob Smith</a>
27    </div>
```

Como puede observarse, la entidad ha de describirse completamente dentro de la etiqueta contenedora marcada con el atributo *itemscope*. A continuación el atributo *itemtype* marca el tipo de entidad que vamos a describir (en este caso, una persona). Después de estas dos declaraciones, la etiqueta puede tener todo el código que sea necesario (en ocasiones en las que se usa la entidad *webpage*, puede contener la página entera). Dentro de ella se marcarán las propiedades de la microdata con el atributo *itemprop*. Así, podemos observar cómo se describe el nombre de la persona con el atributo *itemprop="name"*, así como otros valores, como una imagen o en qué trabaja. Es importante destacar que la microdata permite describir otras entidades dentro de la entidad principal, siempre que ésta esté relacionada con la entidad principal. En caso contrario se debe usar el atributo *itemref* para describir propiedades externas, si bien no es una práctica común. En el caso de arriba vemos cómo la propiedad *itemprop="address"* tiene como valor la clase *postaladress*, que se enuncia con su *itemscope* e *itemtype*, estando anidada dentro de la clase principal: *person*.

Como ya hemos mencionado, ciertas propiedades como *itemprop="url"* no tienen el valor de la propiedad en el propio texto, sino en otros atributos, como en este caso *href*.

De esta forma, usando los tres atributos principales y el amplío vocabulario descrito por *schema.org*, se puede incorporar de forma sencilla información semántica a una página web. A continuación se comentarán algunas herramientas y recursos que ayudan a la construcción de sitios web con microdata.

2.2.3. Herramientas y recursos de microdata actuales

Si bien la implementación de la microdata dentro del código de una página web es sencilla, actualmente existen diversas herramientas que ayudan a los desarrolladores web a crear su información con microdata. Entre ellos destaca la herramienta de verificación de datos estructurados de *Google* [21]. En la figura 2.9, podemos ver cómo la herramienta extrae las propiedades de la persona y su dirección.

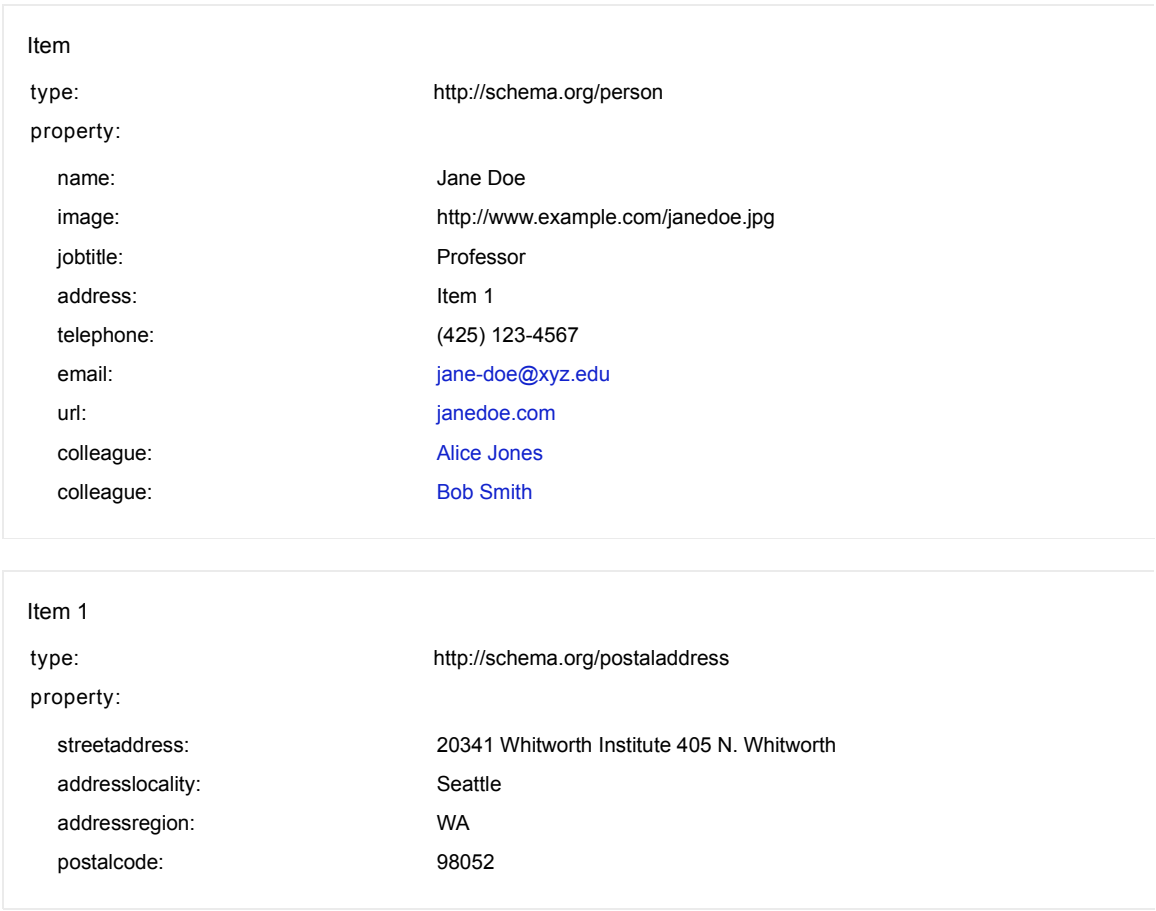

Figura 2.9: Resultados del ejemplo de código con microdata tipo *person* extraídos por la herramienta *Google Structured Data Testing Tool*.

Existen, a su vez, varias herramientas que permiten crear código de microdata, como *microdata-generator.com* o *schema-creator.com*. También existen plantillas como la de *Drupal* [22], que ayudan a la escritura de la microdata. Si bien todas estas herramientas permiten simplificar el proceso de escribir microdata, son en muchos aspectos muy limitadas y no son compatibles con algunas entidades. Por este motivo, siempre es recomendable escribir la microdata directamente en el código *HTML*.

2.2.4. Ventajas y desventajas respecto a otros vocabularios

La microdata supone un formato más en la competencia entre metaformatos que existe actualmente en la web. Su sencillez y clara estructura respecto a otros metaformatos, como los *microformatos* o *RDFa y RDF lite*, así como el apoyo por los principales motores de búsqueda y su uso en la versión de hipertexto *HTML5*, hacen que su uso esté cada vez más extendido. A continuación, en la tabla 2.1, puede verse una comparativa entre los distintos metaformatos actuales.

	Vocabulario	Complejidad	Procesamiento	Extensibilidad
Microformatos	Bajo	Baja	Baja	Ninguna
RDFa	Alto	Alta	Alta	Alta
RDFa Lite	Medio	Baja	Media	Alta
Microdata	Medio	Baja	Baja	Media

Tabla 2.1: Comparativa entre los distintos metaformatos existentes en la web.

Mientras que los *microformatos* son sencillos de utilizar y su nivel de procesamiento es bajo, están muy limitados tanto por su vocabulario como por la carencia de expansibilidad, siendo imposible la creación de microformatos propios. *RDFa* es sin duda el metaformato más potente, teniendo un gran vocabulario y una extensibilidad muy completa. Pero su complejidad y dificultad de procesamiento hacen que su aplicación sea costosa, requiriendo tiempo y experiencia. En esta competencia entre formatos cabe destacar *RDFa Lite*, que surgió como respuesta a la creación de la microdata, y es una simplificación de *RDFa*, siendo su estructura y funcionamiento un calco de ésta. Sin embargo, aporta la ventaja de no necesitar *itemref* para relacionar diversos objetos, así como la flexibilidad de mezclar vocabularios mediante el prefijo *prefix*, característica del lenguaje *RDF* y sus derivados. La microdata, a cambio, ofrece un menor nivel de procesamiento, siendo más eficiente su análisis por parte de los programas extractores.

Un último factor destacable, que no se presenta en la tabla por ser más subjetivo, es el apoyo que recibe cada formato. Los *microformatos* fueron los primeros metaformatos, y actualmente son los más usados en la web. La adopción de *RDFa* ha sido muy pequeña debido a su complejidad. Si bien los especialistas en web semántica la defienden, los diseñadores web prefieren usar otros metaformatos más sencillos de implementar. Son el caso de *RDFa Lite* y *microdata*. Si bien *RDFa Lite* ofrece ventajas superiores en ciertos aspectos semánticos, en la práctica la *Microdata* está teniendo mayor difusión debido al fuerte apoyo recibido por parte de los motores de búsqueda.

Aunque estos motores aceptan también otros metaformatos, en la práctica están trabajando para mejorar y extender el metaformato de la *microdata*. El uso de metaformatos en una página web permite una mayor riqueza a la hora de ofrecer resultados al usuario. Los motores de búsqueda son capaces de indexar mejor las páginas al obtener información más contextualizada, por lo que su uso deriva en un mejor posicionamiento web y en una mayor riqueza en la muestra de los resultados.

Esta riqueza es lo que se denomina como *rich snippet*, y permite a motores como *Google* mostrar mayor información de una web determinada (figura 2.10).

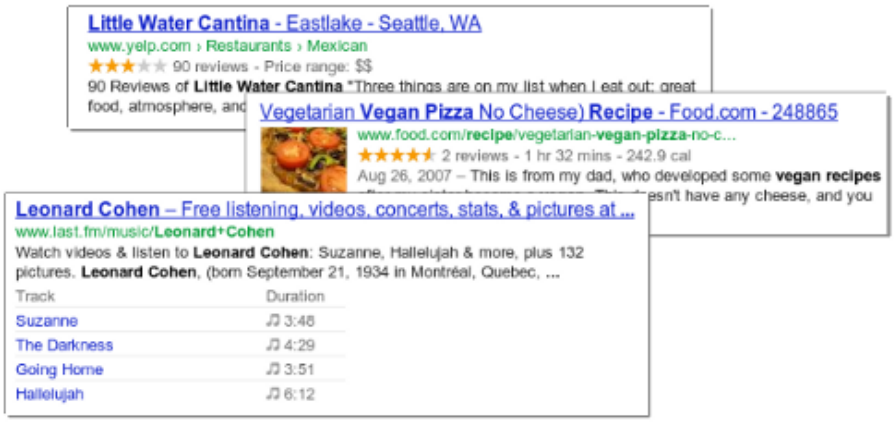

Figura 2.10: Ejemplo de resultados enriquecidos, Google, 2013

2.2.5. Aplicaciones presentes y futuras

La microdata nació en el año 2011 y está íntimamente ligada al formato web *HTML5*. A pesar de su juventud, el gran apoyo de los motores de búsqueda y su utilidad como descriptor semántico está haciendo que se extienda rápidamente. Existen numerosos ejemplos actuales de microdata aplicados en diversos sitios web. El más famoso es sin duda el visor de recetas de *Google*: *Google Recipes View* [23] (solo disponible en EEUU y Japón) que mediante el uso de microdata permite buscar recetas por diversos parámetros, como por ejemplo ingredientes disponibles, tipo de comida o incluso calorías. En la misma línea existen páginas web que aplican la microdata en sus recetas con el fin de mejorar sus búsquedas como *foodnetwork.com* o *yummly.com*.

Otro ejemplo significativo es el sitio *IMDb.com*, Un sitio dedicado a la creación de una gran base de datos de películas, series, actores y otros elementos relacionados con el mundo del espectáculo. *IMDb* utiliza varios recursos de la web semántica como tripletas *RDF* y *microdata*. En el caso de la microdata, se describen elementos multimedia como películas o series, añadiendo información de los actores, críticas y otros parámetros relacionados.

Otro campo en el que se observa actualmente un aumento en el uso de microdata son las webs especializadas en ofrecer imágenes. Bancos de imágenes como *dreamstime.com* o *freefoto.com*, incorporan microdata en sus resultados con el objetivo de ofrecer mayor información sobre las fotos e imágenes que ofrecen, y por tanto mayor precisión en sus búsquedas. Cabe también destacar sitios web pertenecientes a medios de comunicación que poco a poco van incorporando microdata en la descripción de sus artículos, vídeos y fotografías. Un ejemplo de ello es *theguardian.com*, que suele comentar sus artículos, así como describir su autor y sus fotografías, con microdata. También des-

tacan sitios como *rtve.es*, web con un gran archivo multimedia de vídeo y audio descritos mediante microdata, o el grupo *AtresMedia*, que la utiliza para describir noticias y elementos multimedia.

2.3. Arañas web: Herramientas para la obtención de información en la web

Uno de los objetivos de la web semántica es promover el denominado *Internet de las cosas*. La comunicación entre máquinas es habitual desde el nacimiento de Internet y ha ido creciendo hasta llegar al momento actual, en que la navegación por parte de elementos no humanos ha llegado al 61.5 % [24]. Como puede observarse en la figura 2.11, el uso de *bots* en la web es ya superior al humano. Muchos de esos *bots* son los denominados como *rastreadores, arañas web o crawlers*.

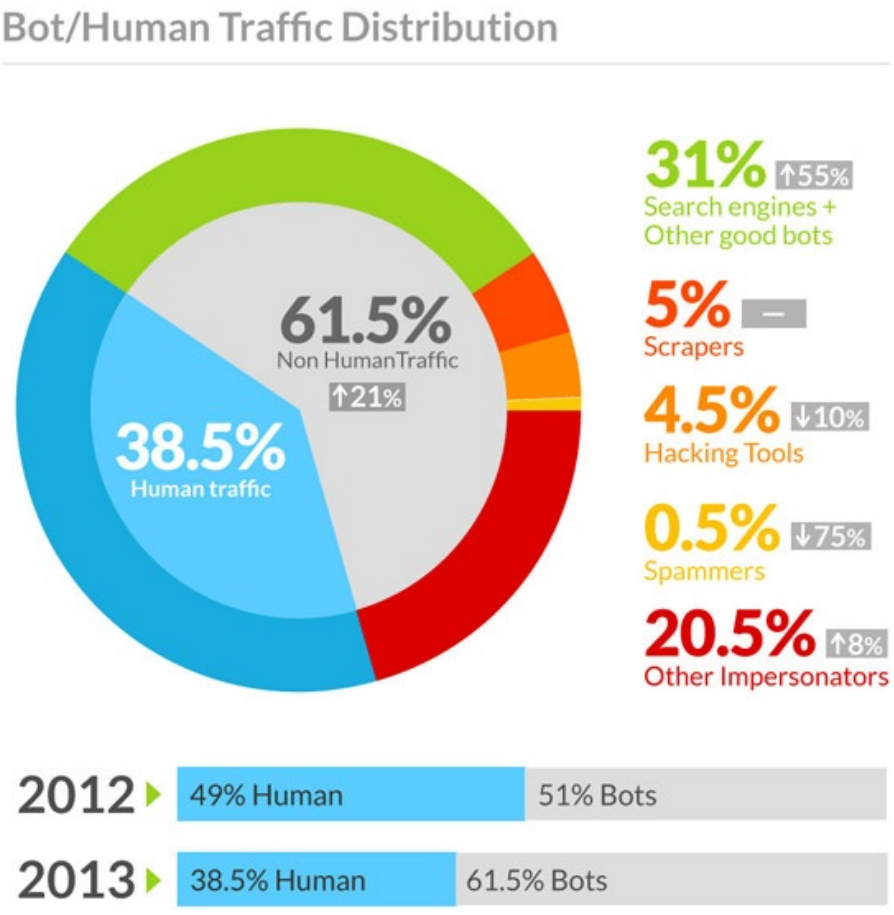

Figura 2.11: Resumen de los resultados obtenidos por *Incapsula Inc*, 2013, relativos al tráfico humano y no humano en el año 2013.

2.3.1. Descripción general y funcionamiento

Un *Web crawler* es un programa diseñado para explorar páginas webs de forma automática. Sus usos van desde la indexación de los sitios web, como *Googlebot*, a tareas de mantenimiento de dominios web como búsqueda de links rotos o la recolección de un tipo de información determinada.

El funcionamiento y estructura de un *crawler* suele dividirse en cuatro partes: *sembrado, filtro, descarga y análisis*, o tres, dependiendo de si el *crawler* está diseñado para analizar las webs encontradas o solo para descargarlas.

Normalmente un *crawler* parte de una lista de direcciones *URL* a analizar, conocidas como *feeds* o semillas. A continuación, el *crawler* realiza un filtrado de las páginas web que le interesan o no según una serie de criterios conocidos como *crawling policies* que dependen de los objetivos del *crawler*, y que pueden pueden ir desde que la página haya sido ya visitada o no, a tipos especiales de páginas, dominios concretos, búsqueda de páginas con formatos específicos, última vez que se visitó la página, etc. Una vez seleccionados los *feeds* que se desea analizar, el *crawler* realiza la descarga de las páginas web y si el *crawler* está diseñado para ello, el análisis, extracción y almacenamiento del contenido buscado.

En la figura 2.12 puede observarse la estructura básica que normalmente usa un *crawler*. Dependiendo de sus objetivos, almacenará o no toda la página web o solo una parte. En el caso de *Googlebot* por ejemplo, el *crawler* solo descarga la página y la prepara para que otro programa externo, cuyo algoritmo protege *Google* como secreto empresarial, realice la indexación del contenido preparándolo para su consulta.

Si bien implementar un *crawler* básico puede ser relativamente sencillo, desarrollar un *crawler* con altos niveles de eficiencia, velocidad y fiabilidad conlleva un desarrollo y perfeccionamiento de dificultad elevada. Debido a ésto, los algoritmos y arquitectura de los principales *crawlers* son celosamente guardados en secreto por parte de las corporaciones. Las arañas web trabajan en el, probablemente, sistema más abierto que existe de información en el mundo: Internet. Se estima que actualmente los *crawlers* solo son capaces de analizar un 15 % de toda la información existente en la web.

Las dificultades a las que puede enfrentarse un *crawler* pueden llegar a ser muy numerosas: duplicidad o excesiva información, redireccionamientos, páginas vacías, código *HTML* erróneamente escrito, que puede resultar un problema para su *parser*, incluso las denominadas como *spider traps* que, creadas de forma intencional o accidental, pueden llevar a un *crawler* a un bucle infinito mediante redireccionamientos, páginas dinámicas, o información a analizar que sea capaz de llevar a un error del programa al *parser* que las analice.

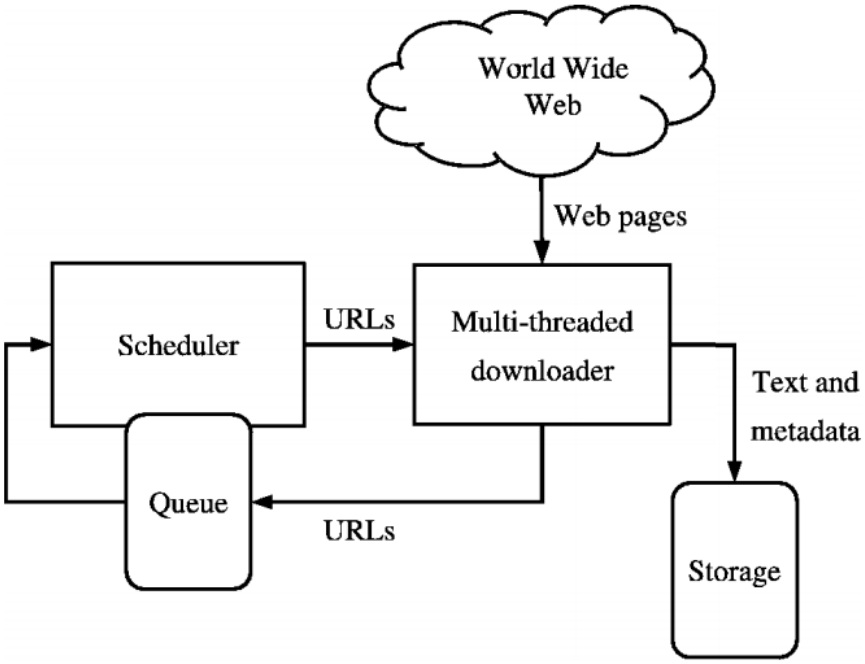

Figura 2.12: Arquitectura general de un *web crawler*, Carlos Castillo, 2004.

2.3.2. Aplicaciones actuales

Las aplicaciones que puede tener una araña web son muy variadas y en la practica infinitas. Como se observaba en la figura 2.11, los *web crawlers* son cada vez más usados por numerosas empresas y organismos para fines comerciales y/o de investigación. Su aplicación más famosa es su uso en los motores de búsqueda. Los *crawlers* son los mecanismos gracias a los que puede indexarse la web, pudiendo realizar búsquedas que arrojen resultados fiables. *Googlebot* y *Yahoo Slurp* son algunos de los *crawlers* más conocidos que día a día analizan la web buscando cambios y nuevo contenido. Cabe destacar también *crawlers* con fines ilegales denominados como *spambots*. Normalmente se basan en la suplantación de identidad online para llevar a cabo operaciones fraudulentas.

Sin embargo, el uso de *crawlers* va mucho más allá de su empleo en los motores de búsqueda. Existen numerosos *crawlers* cuyo objetivo es el mantenimiento de un determinado dominio mediante la comprobación de enlaces rotos, páginas caídas y un sinfín de tareas posibles. Además, son cada vez más usados por parte de empresas para la monitorización de la competencia. Es habitual la generación de informes de comparativas de precios en sectores donde la competencia es muy elevada mediante el uso de *crawlers* que analizan la información encontrada en la web de los competidores de forma periódica. En general, el objetivo primordial de una araña web es analizar la red en busca de una información determinada. Por esta razón su uso en la web semántica cobra vital importancia como medio para obtener información y analizar datos semánticos ricos en contexto. Un ejemplo de ello es *Slug* [25], un *crawler* semántico especializado en la obtención de tripletas *RDF*. Como se ha

podido observar, el uso de *bots* en Internet está creciendo cada vez más. El desarrollo de dispositivos móviles, así como de nuevos elementos electrónicos conectados a Internet mediante el denominado *Internet de las cosas*, augura un crecimiento aún mayor en el que la comunicación entre máquinas y la traducción de la información al lenguaje natural humano cobrarán cada vez más importancia. Por este motivo la web semántica es fundamental para el futuro desarrollo de la red de redes que, cada vez más, nos mantiene conectados como una sola inteligencia colectiva.

Capítulo 3

Descripción del programa *Onewebmicrodata*

En este capítulo se describirá el programa *Onewebmicrodata*. Se trata de un programa tipo *crawler* cuyo objetivo es la búsqueda, recolección, análisis y extracción de imágenes con información semántica, en formato de microdata, presentes en la web. Se describirán, por tanto sus objetivos y requisitos, así como su estructura y otros aspectos presentes en la creación de un programa rastreador de estas características.

3.1. Descripción general

El programa *Onewebmicrodata* es un programa del tipo *araña web* o *crawler*. Estos programas están enfocados a la exploración de páginas web con distintos propósitos, desde tareas de mantenimiento en dominios a la búsqueda y extracción de una determinada información. En el caso del *crawler* realizado, su objetivo es la localización y extracción de imágenes con microdata presentes en la web.

La microdata, tal como se explicó en el apartado 2.2, es un tipo de metadato presente en páginas escritas en *HTML5* cuyo objetivo es la descripción semántica de distintas entidades. Con ella se pueden describir de una forma detallada conceptos como *personas, empresas, productos, películas, etc.*, mediante el uso de clases y atributos, propios de cada concepto, incrustados en el código *HTML*.

En el caso del presente programa se pretende localizar y extraer la información que describa imágenes mediante microdata presentes en la web. Para la descripción de imágenes existen dos clases de microdata a tener en cuenta: *ImageObject* [26] y *Photograph* [27]. La primera describe cualquier tipo de imagen presente en la web, mientras que la segunda está centrada en la descripción de fotografías. Estas dos clases son hijas de otras más generales de las que heredan sus atributos,

así como padres de otras clases anidadas o subclases que pueden ser descritas dentro de éstas (por ejemplo, la persona que ha realizado la fotografía). Para ello el programa se divide en dos partes, la parte del *crawler* o exploración, encargada de moverse de web en web descargando y procesando el código *HTML*, y la parte de extracción y análisis, encargada de la extracción y lectura de la microdata.

Así pues, el programa partirá de una página web o conjunto de páginas web especificado, analizando el código de cada una y moviéndose a la siguiente. En el caso de que encuentre imágenes con microdata, el programa descargará la imagen y extraerá la información de ésta en un documento *txt* del mismo nombre que la imagen.

A continuación se describirán los objetivos generales del programa, así como sus especificaciones de diseño y arquitectura. Finalmente se presentará una serie de ensayos realizados con el programa junto con sus resultados.

3.2. Especificaciones de diseño

El programa busca cumplir una serie de objetivos generales que se especifican a continuación:

3.2.1. Objetivos generales y requisitos del sistema

- **Exploración web:** El programa debe ser capaz de analizar una página web, moviéndose posteriormente y de forma automática a una nueva página vinculada a la anterior. Debe ser capaz de repetir este proceso reiteradamente con el objetivo analizar el mayor número de páginas web posible.

- **Identificación de imágenes descritas mediante microdata:** El programa debe ser capaz de analizar el código de cada página web que explore, identificando si existen o no imágenes con microdata dentro de ésta.

- **Extracción de las imágenes:** Si el programa detecta que existe una imagen con información en forma de microdata, debe ser capaz de extraer la imagen descargándola al ordenador.

- **Análisis y extracción de la microdata adjunta:** El programa debe ser capaz de analizar la microdata asociada a la imagen, extrayendo su información a un archivo de texto plano *txt*, cuyo nombre debe ser el mismo que el de la imagen descargada.

- **Extracción y análisis de datos Exif:** Los datos *Exif* son información incrustada dentro del binario de un archivo tipo imagen *JPEG o TIFF*, no siendo soportado por formatos como *JPEG 2000* y *PNG*. Los datos *Exif* son, normalmente, generados a partir de la realización de una fotografía o la manipulación de una imagen mediante un programa de edición. El objetivo

de los datos *Exif* es almacenar información referente a la fotografía, como el modelo de cámara usado, la apertura, velocidad de obturación, sensibilidad o *ISO*, uso del flash, etc. El programa debe ser capaz de analizar si las imágenes descargadas poseen o no datos *Exif* extrayendo y añadiendo dicha información al archivo *txt* con su mismo nombre. A su vez, el programa debe ser capaz de, en función de los parámetros obtenidos por los datos *Exif*, extraer conclusiones sobre el tipo de fotografía que se ha realizado (fotografía de noche, retrato, paisaje, macro, etc).

Para alcanzar estos objetivos, el programa deberá cumplir una serie de requisitos mínimos:

- El programa deberá ser capaz de realizar la descarga de archivos presentes en la red, tanto documentos *HTML*, como imágenes.
- A su vez, el programa deberá ser capaz de identificar otro tipo de archivos, evitando su descarga.
- El programa deberá ser capaz de analizar el código *HTML* obtenido de cada página web descargada, identificando y/o modificando, si es necesario, el contenido de dicho código, así como los enlaces *URL* presentes en éste.
- El programa deberá ser capaz de, a partir de una página web dada, moverse a otra página web para realizar un nuevo análisis.
- El programa deberá identificar y extraer el código que contenga imágenes descritas mediante microdata.
- El programa deberá analizar dicho código con microdata, descargando la imagen que describe y extrayendo a un fichero de texto la información sobre ésta.
- El programa deberá ser capaz de identificar qué imágenes descargadas poseen datos *Exif*, extrayendo dichos datos junto la información extraída de la microdata.

A continuación se describirán los aspectos del programa claves para alcanzar los objetivos y requisitos mencionados: lenguaje de programación, librerías y estructura básica de funcionamiento.

3.2.2. Lenguaje de programación utilizado

Para la realización del programa se ha usado el lenguaje de programación *C++*.

Su elección frente a otros lenguajes de programación viene dada por diversas razones.

En primer lugar ha influido la cantidad de librerías disponibles en *C++*. Si bien algunas librerías fundamentales del programa, como *libcurl*, disponen de versiones para prácticamente cualquiera de los lenguajes actuales, otras librerías, para según qué tareas, pueden ser más difíciles de hallar. Esta

variedad, es de gran importancia en un programa tipo *crawler* o araña web, cuya funcionalidad puede ser adaptada a multitud de propósitos.

En segundo lugar, tanto para el análisis del código *HTML* de la parte de *crawler* del programa, como para el *parser* de microdata, la velocidad de procesamiento del lenguaje *C++* respecto a otros lenguajes, sobre todo interpretados como *Java* o *Python*, es, en general, significativamente superior. En un programa de este tipo, donde se pretende analizar grandes volúmenes de información, es fundamental intentar optimizar todo lo posible la velocidad del programa. Tal como se comentaba en el apartado 2.3, las bases de funcionamiento de un programa *crawler* son relativamente sencillas. La dificultad estriba en la habilidad de dicho programa para afrontar sucesos inesperados mientras analiza la web y ser lo más eficiente y rápido posible. En la actualidad los *bots web* solo son capaces de analizar un 15 % del contenido presente en la web. Una mayor optimización y velocidad implica de forma directa un volumen mayor de páginas analizadas y, por tanto, mayores posibilidades de cumplir el objetivo del *crawler*, en este caso la extracción de imágenes con microdata.

Por último, se ha optado por el uso de *C++* frente al lenguaje *C*, dada la posibilidad de la programación orientada a objetos que ofrece. Si bien en el programa no se ha implementado, debido a que el objetivo era solo la extracción de microdata, esta programación orientada a objetos se adapta muy bien a éste concepto, en el que las distintas clases se relacionan entre sí de forma jerárquica. Si en un futuro se desea que el programa no solo extraiga, sino que además manipule, relacione e infiera nueva microdata a partir de la existente, *C++* permitirá de una forma sencilla la implementación de las nuevas clases creadas con el código ya existente.

3.2.3. Librerías externas utilizadas

Aparte de las librerías estándar de *C++*, se han utilizado las siguientes librerías externas durante el desarrollo del programa:

- **libcurl:** Para la descarga de archivos desde la web, se ha usado la librería *libcurl* [28], una potente librería gratuita y multiplataforma adaptada a multitud de lenguajes cuyo objetivo es la trasferencia de archivos. Es ampliamente usada para la carga y descarga de archivos mediante el protocolo *HTTP* y *FTP*.

  Existen múltiples librerías de transferencias de archivos, tales como *libghttp* [29], cuyo funcionamiento es similar a *libcurl*. Otra buena opción es *libwww* [30], más difícil de manejar, pero que ofrece a cambio el proceso denominado como *caching*, donde el gestor puede almacenar en la memoria caché parte del archivo descargado, de forma que su acceso en las siguientes ocasiones sea más rápido y eficaz.

  En sus últimas versiones, *libcurl* destaca por la posibilidad de dos métodos de programación: *easy_interface* y *multi_interface*. La principal diferencia entre ambas es que la interfaz múltiple

(*multi_interface*) permite un absoluto control de los parámetros de transferencia, pudiendo realizarse la transferencia de varios archivos a la vez basándose en la estructura básica o *easy_interface*. En cambio la interfaz simple *easy_interface*, permite una programación más sencilla y directa, siendo la opción más conveniente para la mayoría de las tareas a realizar.

En el desarrollo de este programa se ha usado la interfaz básica, o *easy_interface*, para la descarga de archivos de dos tipos: páginas *HMTL* e imágenes.

- **tinyxml:** Para el análisis o *parseo* de las etiquetas *HMTL* que contienen microdata se ha usado la librería *tinyxml* [31]. *Tinyxml* es una librería especializada en el análisis de documentos *XML*. Destaca frente a otras librerías similares como *libxml2* [32] o *irrXML* [33] por su sencillez de uso y rapidez de procesamiento.

  Actualmente existe una segunda versión de *tinyxml* denominada como *tinyxml2* [34]. En muchos aspectos, *tinyxml2* promete convertirse en un *parser* con mayor eficiencia en el uso de la memoria, así como mayor ligereza y con una mejor gestión de los espacios en blanco del documento *XML*. Sin embargo, *tinyxml2* aún está en un desarrollo temprano, siendo *tinyxml* una librería más madura cuya principal ventaja respecto a su sucesor es la posibilidad de reportar errores encontrados durante el análisis del documento *XML*. Esta posibilidad de reportar errores, opción fundamental a la hora de desarrollar y mejorar un *parser* que trabaja en un sistema abierto como es la web, unido a un código ya en su completo desarrollo, ha hecho que para este programa se utilice la primera versión de la librería, si bien en un futuro podría adaptarse para su uso con la segunda versión.

- **easyexif:** Para el análisis de los datos *Exif* presente en numerosas imágenes, así como su extracción, se ha usado la librería *easyexif* [35].

  Para el lenguaje de programación *C++*, hay varias librerías capaces de manipular datos *Exif*. Una de ellas son *exiv2* [36], librería que permite extraer, manipular y reescribir distintos tipos de metadata presente en imágenes. También cabe destacar *libexif* [37]. Sin embargo en el programa se ha optado por usar la librería *easyexif*. Esta librería solo permite la lectura y extracción de los datos *Exif*, no pudiendo sobrescribir ni modificar dichos datos presentes en el archivo binario de la imagen. Sin embargo, y dado que el objetivo del programa es la extracción de dichos datos y no su manipulación, es una gran opción debido a su sencillez y ligereza.

3.2.4. Estructura básica del programa

A continuación, en la figura 3.1 se muestra la estructura básica del programa:

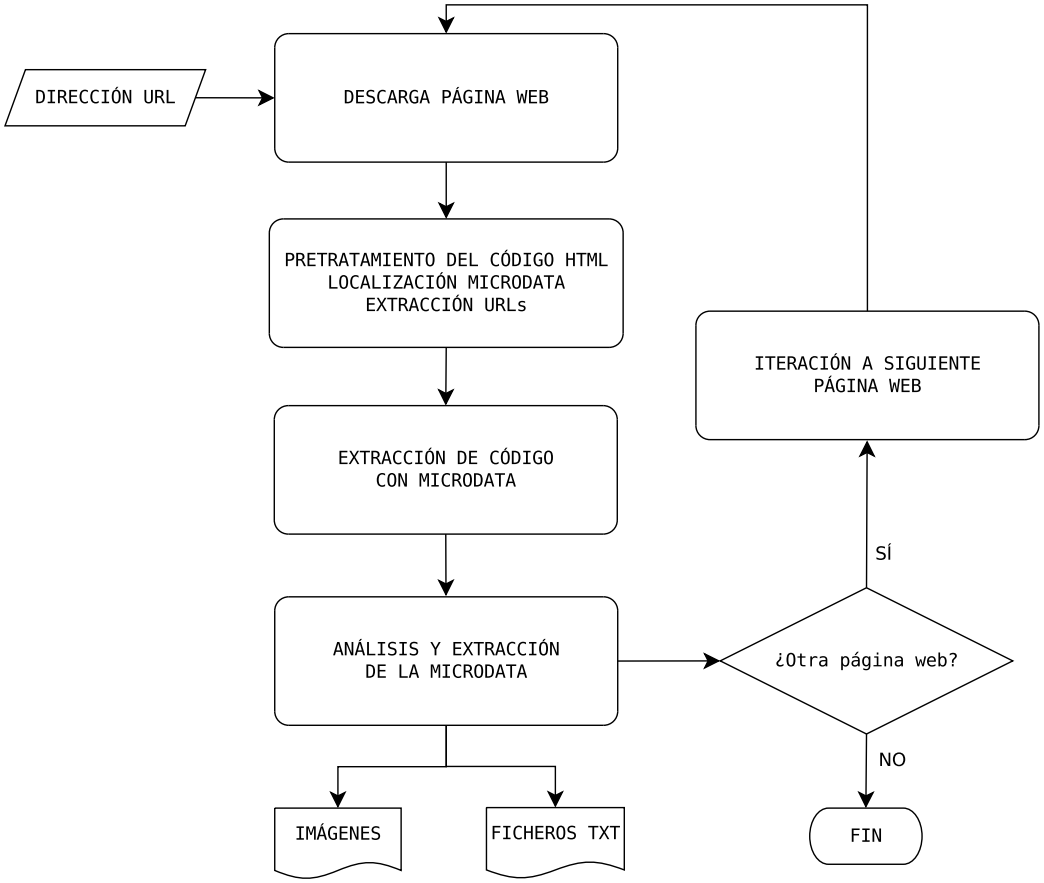

Figura 3.1: Estructura general del comportamiento del programa. El programa está dividido en cuatro grandes bloques que se repiten en cada página web visitada.

Como se puede observar, el programa está dividido en cuatro grandes bloques, siendo su funcionamiento básico el siguiente:

1. En primer lugar el programa recibe una dirección *URL* para analizar.

2. El programa descarga el código *HTML* en el bloque *Descarga página web*.

3. En el segundo bloque, el programa lleva a cabo un pretratamiento del código *HTML* que facilitará la extracción y análisis posterior de las etiquetas con microdata. Además, este bloque se encarga de la localización de etiquetas con microdata así como de la extracción de las direcciones *URL* a las que apunta la página web descargada.

4. En el tercer bloque, el programa identifica el tipo de microdata localizada por el bloque anterior, seleccionando y extrayendo solo la microdata asociada a las imágenes que se desea.

5. En el último bloque: *Análisis y extracción de microdata*, el programa analiza las etiquetas *HTML* extraídas del código anterior que contengan microdata, descargando la imagen descrita y generando un fichero de texto plano con la información extraída de ésta.

6. Por último, el programa selecciona una de las *URL* a las que apuntaba la página web descargada, repitiéndose todo el proceso el número de veces deseado.

En el siguiente apartado se comentará de forma más detallada el funcionamiento de cada bloque.

3.3. Definición de la arquitectura

A continuación se comentará con mayor detalle el funcionamiento de cada uno de los bloques de funcionamiento del programa (figura 3.1):

3.3.1. Bloque 1: Descarga página web

El objetivo de este bloque de programación es la descarga de páginas web al ordenador para su posterior análisis. En la figura 3.2 puede verse la estructura de funcionamiento del bloque.

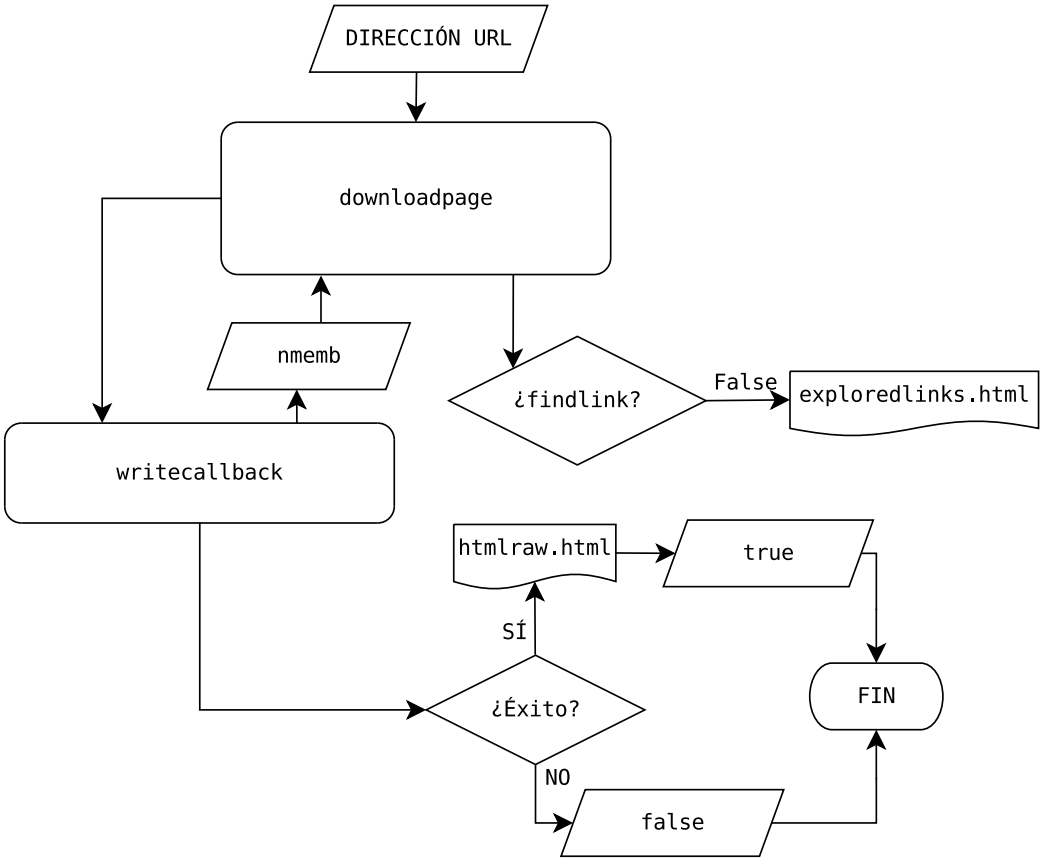

Figura 3.2: Estructura de funcionamiento del bloque principal: *Descarga página web.*

En el bloque 1 *Descarga página web* se recogen todas las funciones de descarga de archivos, tanto imágenes como documentos *http*, así como las funciones auxiliares que facilitan dichas descargas. A continuación se comentará el funcionamiento y puntos clave necesarios para la descarga de una página web:

- **downloadpage:**

  Este bloque recibe dos parámetros para funcionar. El primero es la dirección *URL* que se desea descargar. El segundo permite activar o desactivar la información sobre la conexión y descarga que *libcurl* muestra por pantalla. Esta información puede ser muy útil, tanto para la mejora del programa como para averiguar el estado actual de una página web. En el caso de que se desee mostrar la información de descarga, el programa mostrará algo como esto:

Ejemplo de información de descarga mostrada por el programa.

```
1  Descargando codigo de la url: http://www.uc3m.es/Inicio
2  * About to connect() to www.uc3m.es port 80 (#0)
3  *   Trying 213.134.43.166...
4  * Connected to www.uc3m.es (213.134.43.166) port 80 (#0)
5  > GET /Inicio HTTP/1.1
6  Host: www.uc3m.es
7  Accept: */*
8
9  < HTTP/1.1 200 OK
10 < Server: Apache-Coyote/1.1
11 < Set-Cookie: JSESSIONID=F6E6C78520E3D91C0099012E9E743246; Path=/; HttpOnly
12 < Cache-Control: no-store
13 < Set-Cookie: SS_X_JSESSIONID=62D5ECA91F05D5967CA33E2DA7B87FC2; Path=/
14 < vary: Accept-Encoding
15 < Last-Modified: Fri, 03 Jan 2014 17:19:56 UTC
16 < Content-Type: text/html;charset=UTF-8
17 < Transfer-Encoding: chunked
18 < Date: Fri, 03 Jan 2014 17:19:56 GMT
19 <
20 * Connection #0 to host www.uc3m.es left intact
```

Como puede observarse, el programa muestra por pantalla datos como a qué puerto se conecta, la *IP* del dominio, versión de *HTTP*, así como nombre del servidor, cookies utilizadas o incluso el contenido del archivo a descargar.

Funcionamiento: Este bloque del programa usa dos archivos. El primero: *htmlraw.html* será donde se almacenará el código de la página web descargada. El segundo: *exploredlinks.html* hace la función de historial de navegación del programa, donde se guardarán todas las *URL* ya visitadas.

Antes de llevar a cabo la descarga del archivo se debe comprobar si la *URL* de la página que se desea descargar está codificada o no, como puede verse en el siguiente ejemplo:

URL sin codificar: http://www.uc3m.es/Inicio

URL codificada: http%3a%2f%2fwww.uc3m.es%2fInicio

Las direcciones *URL* utilizan el código *ASCII*. En ocasiones ciertos caracteres, como *:, =, & y /* así como otros caracteres especiales, o bien están reservados o pueden inducir a error. Por ello en algunas ocasiones una *URL* se transmite de forma codificada sustituyendo sus caracteres conflictivos por su notación en código hexadecimal, siendo necesario descodificarla para iniciar la descarga.

En el caso de que la *URL* esté codificada, el programa realiza la descodificación antes de proceder a la descarga.

A continuación se procede a establecer los parámetros de la librería *libcurl* que definirán el comportamiento del bloque a la hora de realizar una descarga. De estos parámetros caben destacar *CURLOPT_FOLLOWLOCATION* y *CURLOPT_MAXREDIRS*. El primero establece que, en el caso de que la página a descargar esté redireccionada a otro lugar, siga automáticamente esta redirección. El segundo establece el número máximo de redirecciones a seguir. Se pretende con esto último evitar grupos de páginas que se redireccionan de forma recíproca, ya que supondrían un bucle infinito en la navegación.

En ocasiones, al intentar descargar una página web, *libcurl* consigue abrir la conexión pero el servidor no da respuesta, permaneciendo el programa a la espera de ésta un tiempo indeterminado. Para evitar esta situación el programa usa el parámetro *CURLOPT_CONNECTTIMEOUT*, que establece el tiempo máximo en el que se mantiene abierta la conexión. Se ha establecido un tiempo máxmo de *600* segundos, es decir, un tiempo máximo de diez minutos. Si pasado este tiempo la página web continúa sin dar respuesta, el programa cierra la conexión y prosigue su ejecución.

Si el servidor da respuesta, se procede a descargar la página web, volcando su información al archivo *htmlraw.html*. Además, el bloque *findlink* comprueba si la *url* existe o no en el archivo de historial *exploredlinks.html*, añadiéndose a éste en caso negativo. En caso de que todo vaya bien, el bloque devolverá el valor *true*, indicando que la descarga se ha realizado con éxito.

- **writeCallback:** Bloque utilizado por la librería *libcurl* para escribir el archivo descargado y calcular su tamaño. En caso de que el archivo final tenga un tamaño distinto al previsto, el bloque sabrá que la descarga no se ha realizado con éxito.

- **findlink:** Este bloque se encarga de comprobar si el historial de navegación del programa, contenido en el archivo *exploredlinks.html*, dispone ya o no de la *URL* descargada, devolviendo *true* en caso afirmativo y *false* en el caso contrario.

3.3.2. Bloque 2: Análisis del *HTML*

El objetivo de este bloque puede dividirse en tres partes:

- **Pretratamiento del código *HTML*:** Analiza el código *HTML*, modificando y eliminando código que más tarde será innecesario o puede suponer un problema para el análisis de las etiquetas que contengan microdata.
- **Localización del código con microdata:** Localiza las etiquetas que contienen información con microdata, escribiendo en *itemscope* la *URL* donde se ha encontrado dicha microdata. Ésta será extraída más tarde por el *Bloque 3: Extracción de microdata* (apartado 3.3.3).
- **Extracción de vínculos *URL*:** Cuando localiza un vínculo *URL* a otra página, lo extrae almacenándolo en el archivo *href.html*. Éste, posteriormente, servirá para descargar y analizar una nueva página.

En la figura 3.3 se presenta un diagrama del funcionamiento básico del bloque. Al final de éste se habrán generado tres archivos: *htmlprocessed.html* y *href.html* que se usarán para los bloques posteriores, así como *foundmicrodata.html*, un archivo destinado al *debug* del programa que almacenará las *URLs* donde se ha encontrado imágenes con microdata.

En el bloque 2 *Análisis de HTML* se recogen todas las funciones que permiten realizar las tres partes anteriormente comentadas. siendo su bloque principal *processhtml*. Ésta llamará, dependiendo de los parámetros seleccionados en el programa, a distintos bloques auxiliares que permitirán mejorar los archivos generados o realizar acciones secundarias, como la descarga de imágenes sin microdata.

A continuación se comentará el funcionamiento y puntos clave necesarios para el correcto funcionamiento de ésta parte del programa.

- **processhtml:** Es el bloque principal de esta parte del programa. Es altamente configurable en su comportamiento. Además, aporta información esencial al resto del programa.

  Funcionamiento: En primer lugar, el bloque crea dos archivos: *htmlprocessed.html* y *foundmicrodata.html*. El primer archivo almacenará las modificaciones que se han realizado en el archivo *htmlraw.html*, archivo donde se había escrito previamente el código *HTML* de la página a analizar. El segundo archivo tiene el único propósito de almacenar de forma incremental las *URLs* donde se ha encontrado en algún momento la microdata tipo imagen que se busca, siendo un archivo útil para volver a dichas páginas y facilitar la depuración del programa.

  A continuación se procede al análisis y modificación del código *HTML* contenido en el archivo *htmlraw.html*. Para ello el programa leerá de forma individual cada línea del archivo, modificándolas en caso necesario. Para el análisis del código el programa utiliza el siguiente método: en primer lugar se declaran las palabras o sentencias clave que se desean localizar en cada línea

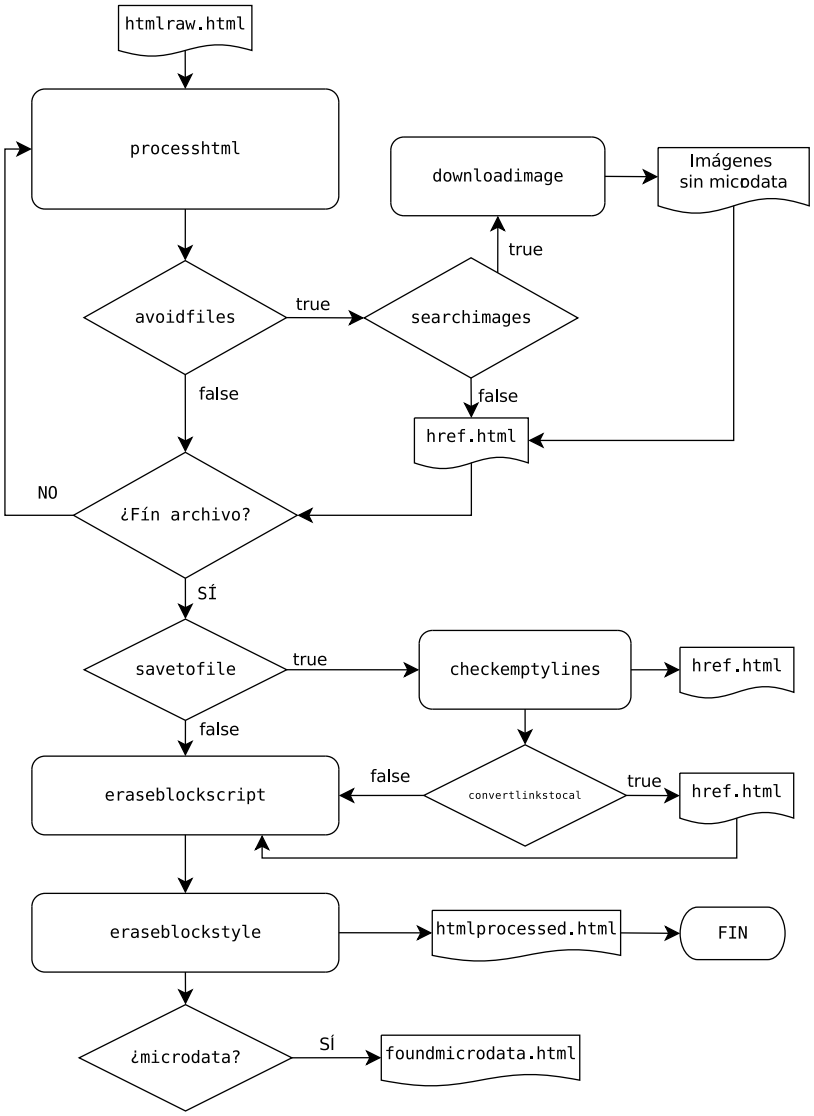

Figura 3.3: Estructura de funcionamiento del segundo bloque principal. En él se analiza el código *HTML* descargado del bloque anterior, modificándolo y extrayendo información del mismo.

del archivo. A continuación una serie de sentencias condicionales comprueban en cada línea si hay coincidencias o no, realizándose las acciones que se deseen en caso afirmativo.

De este modo, mediante la declaración de palabras clave y el uso de sentencias condicionales, se puede modificar y extraer la información necesaria de cada línea del código *HTML*. Esta forma de analizar el archivo presenta la ventaja de ser fácilmente manipulable, obteniéndose al final un listado de palabras clave y acciones a realizar fácil de ampliar y/o modificar.

Entre la información que se pretende extraer están las *URLs* que direccionan a otras páginas web. Debido a que en muchas ocasiones las *URL* no dirigen a una página web, si no a archivos de varios tipos tales como documentos *PDF*, imágenes *ISO* y otros archivos con distintas

extensiones es necesario verificar que sea un documento valido evitando el resto de archivos. Para ello, antes de almacenar una *URL*, se llama al bloque *avoidfiles*. Éste devolverá *true* en el caso de que el link sea válido, guardándose en el archivo *href.html*.

La edición del código *HTML* es fundamental para el correcto funcionamiento del *Bloque 4: Análisis y extracción de microdata* (apartado 3.3.4). La existencia de etiquetas sin cierre, líneas y bloques de comentarios, o código no *HTML* insertado en etiquetas como *script* o *style*, pueden llevar a la librería *tinyxml* a provocar un error durante el análisis. En este sentido, son especialmente problemáticos los atributos sin valor, ya que llevan a la mala interpretación de la estructura del código a analizar por parte del *parser*, provocando que no se capture de forma correcta la información necesaria. En este caso destaca el atributo de introducción de la microdata *itemscope*. Aunque formalmente dicho atributo no posee valor, yendo directamente seguido de la definición de la clase *itemtype*, en la práctica es recomendable asignarle un valor como *itemscope="itemscope"* o *itemscope=""* para facilitar la lectura del código por parte de *crawlers* y otros bots web. En el caso de que el programa encuentre un atributo *itemscope* al que no se le ha asignado ningún valor, se le asignará como valor la *URL* de la página analizada. Esta *URL* será útil en el caso de que se desee recuperar una *URL* a partir de una ruta incompleta.

Tras analizar todas las líneas del código *HTML* se crea un archivo denominado *htmlprocessed.html*, donde se volcará toda la información del código modificado. En el caso de que la variable *bool savetofile* tenga el valor de *true* se copiará a su vez todas las *URLs* que direccionan a otras páginas web en el archivo *href.html*, además de evitar posibles líneas vacías mediante la función *checkemptylines*. Si se selecciona que se desea convertir las *URL* parciales a *URL* completas (mediante la variable bool convertlinkstolocal==true), se llamará a la función *convertlinkslocal* para su conversión, modificación, y nuevo guardado.

- **avoidfiles:** Este bloque cumple dos objetivos:

  - Comprobar si la *URL* pasada al bloque contiene algunas de las extensiones de archivos que no se desean descargar.
  - En caso de que esté activada la búsqueda de imágenes sin microdata por palabra clave (*searchimages*), proceder a la descarga de éstas.

  Para el primer objetivo, y de forma similar a *processhtml*, se realiza un listado de palabras clave que buscar en la *URL*. Estas palabras clave serán extensiones de archivo que no se desean descargar, tales como *pdf, txt, iso, etc.* En el caso de que la *URL* no contenga ninguna de estas extensiones, el bloque devolverá el valor *true*, indicando de esta forma que la *URL* es válida.

Para el caso de búsqueda de imágenes sin microdata por palabra clave, se comprobará si la *URL* pasada al bloque posee una extensión de imagen (*jpg, png* o *gif*) y su *URL* o su descripción dentro de la etiqueta *img*, contiene alguna de las palabras claves introducidas por el usuario. En el caso de que sea una imagen que interese descargar, el programa transformará su vínculo parcial a completo en caso de que sea necesario, y llevará a cargo su descarga llamando al bloque *downloadimage* cuyo funcionamiento se explicará más adelante.

- **checkemptylines:** Este bloque tiene como objetivo eliminar posibles líneas vacías en el fichero de *URLs*: *href.html*. Para ello comprueba la longitud de cada línea del archivo, ignorando las líneas vacías y volviendo a escribir el resto.

- **convertlinkslocal:** Este bloque convierte las rutas parciales a *URL* completas que puedan ser descargadas. Para ello el bloque reconstruye rutas incompletas como: */Vida_Universitaria* combinándolas con la ruta raíz de su dominio, obteniéndose su ruta completa: *http://www.uc3m.es/Vida_Universitaria*.

- **eraseblockscript y eraseblockstyle:** Estos dos bloques tienen como objetivo eliminar del archivo *htmlprocessed.html* las etiquetas *script* y *style* que ocupen varias líneas, facilitando y mejorando la velocidad de lectura posterior del programa.

El uso de este bloque es fundamental para el posterior buen funcionamiento del programa. En Internet existen numerosas páginas web con errores en su código *HTML* que pueden ser una dificultad a la hora de analizar los fragmentos de código con microdata. A su vez, el analizar línea a línea el código nos permite extraer los enlaces *URL* a los que apunta la páginas, así como aportar flexibilidad para encontrar cualquier nuevo tipo de información presente en el código que se desee, tanto para mejorar las funciones del programa existentes como para adaptar el programa a nuevas opciones.

3.3.3. Bloque 3: Extracción de código con microdata

El objetivo de este bloque es la localización exacta de las etiquetas con microdata en el código *HTML* ya procesado (contenido en el archivo *htmlprocessed.html*) con el objetivo de extraerlas del resto del código para su posterior análisis. En la figura 3.4 se presenta un diagrama de funcionamiento del bloque.

Debido a que *HTML* es un lenguaje descriptivo mucho menos formal que *XML* en su gramática, analizar todo el código de la página *HTML* puede presentar numerosos inconvenientes, así como ralentizar el funcionamiento del programa de una forma significativa. Por este motivo se ha optado por extraer las etiquetas que contengan microdata a partir del código principal. De esta forma, y tras el pretratamiento realizado en el bloque anterior, el *parser* podrá realizar un análisis del código mucho más ordenado y eficiente.

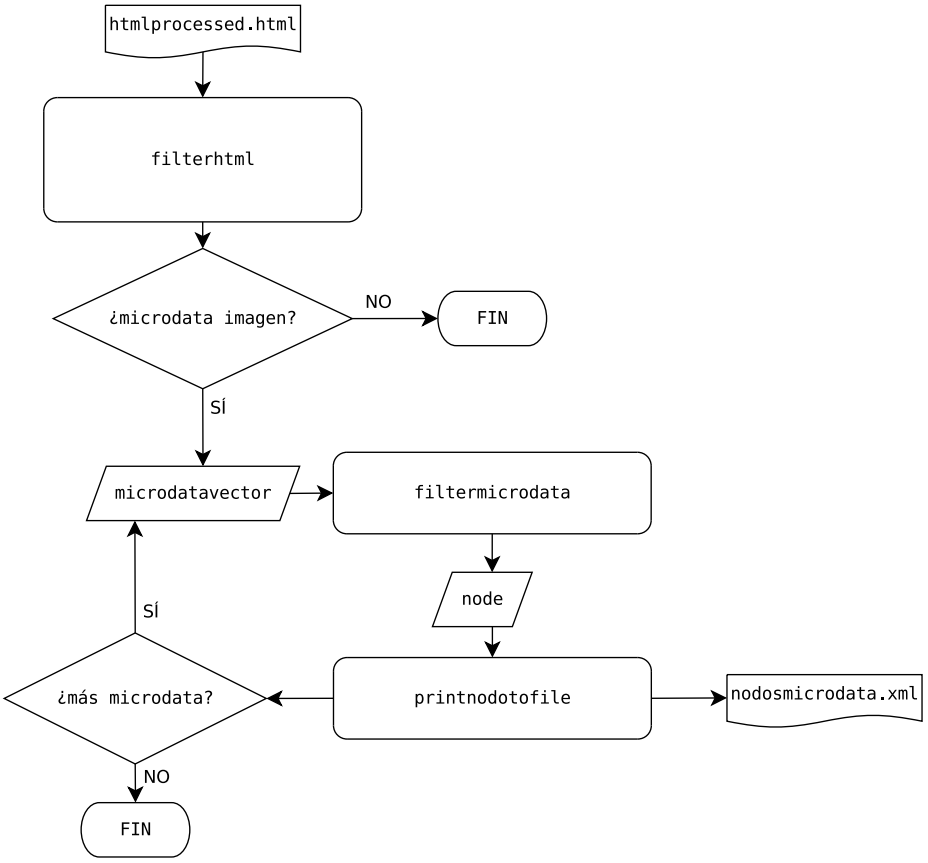

Figura 3.4: Estructura de funcionamiento del tercer bloque principal. El bloque *filterhtml* detecta cada línea de microdata encontrada en el archivo *htmlprocessed.html* llamando a *filtermicrodata* para su extracción.

Este bloque del programa generará como salida el archivo *nodosmicrodata.xml*, donde se almacenarán todos los nodos con microdata extraídos para poder ser analizados por el siguiente bloque (*Bloque 4: Análisis y extracción de la microdata*, apartado 3.3.4). A continuación se comentará el funcionamiento de sus distintas partes:

- **filterhtml:** Este bloque lee el archivo *htmlprocessed* comprobando que exista o no microdata que describa una imagen mediante la lectura del atributo *itemtype*. Actualmente existen dos tipos de microdata tipo imagen que el programa puede extraer:

```
1 <div itemscope="" itemtype="http://schema.org/ImageObject">(...)</div>
2
3 <div itemscope="" itemtype="http://schema.org/Photograph">(...)</div>
```

  En el caso de que se encuentre microdata tipo imagen, el bloque almacenará su posición dentro del archivo, llamando al final de la lectura del archivo al bloque *filtermicrodata* tantas veces como el número de bloques de microdata encontrados.

- **filtermicrodata:** Este bloque recibe por parte del bloque *filterhtml* las posiciones del archivo *htmlprocessed.html* donde se ha encontrado microdata. Éste creará un archivo auxiliar denominado como *filterhttml.html* que almacenará de forma temporal parte del archivo *htmlprocessed.html*, conteniendo solo la información del archivo a partir de la línea donde se encontró microdata. De esta forma resulta mucho más fácil la localización del nodo de microdata que nos interesa, usando los objetos y funciones de la librería *tinyxml* al ser éste el primero del archivo a leer, en vez de tener que recorrer todo el archivo en busca de dicho nodo. Una vez asignado el nodo se pasa éste al bloque *printnodotofile*.

- **printnodotofile:** Este bloque recibe como argumento el nodo que contiene la etiqueta *HTML* con microdata, extrayendo y escribiendo éste en el archivo de salida: *nodosmicrodata.xml*.

Al terminar el bloque se habrá obtenido el archivo *nodosmicrodata.xml*, que contendrá todos los nodos contenedores de microdata hallados en la web analizada. Este archivo será usado por el siguiente bloque del programa para llevar a cabo el análisis de la microdata, así como la descarga de las imágenes descritas.

3.3.4. Bloque 4: Análisis y extracción de la microdata

El objetivo de este bloque es el análisis de los nodos con microdata extraídos previamente por el programa, descargando las imágenes asociadas a dicha microdata y generando un archivo con el mismo nombre que la imagen que contenga la información aportada por la microdata sobre ésta. Para ello se usará la librería *tinyxml*, que permitirá analizar la estructura arbolada de los nodos con microdata almacenados en el archivo *nodosmicrodata.xml*.

Para que el programa extraiga dicha microdata es necesario crear una serie de funciones que sean capaces de entender las clases y atributos que se están describiendo. Dicha información está disponible en la web *schema.org* [19]. Dado que el programa solo se encargará de la extracción de microdata que describa imágenes, éste se centrará en las clases *imageobject* [26] y *photograph* [27], así como en las clases relacionadas con éstas y que, por tanto, pueden aparecer descritas dentro de ellas.

En la figura 3.5 se representa un diagrama con el funcionamiento general del bloque. Este bloque generará finalmente los archivos de salida del programa, es decir, las imágenes a un archivo *txt* con la información extraída en la descripción de microdata de éstas.

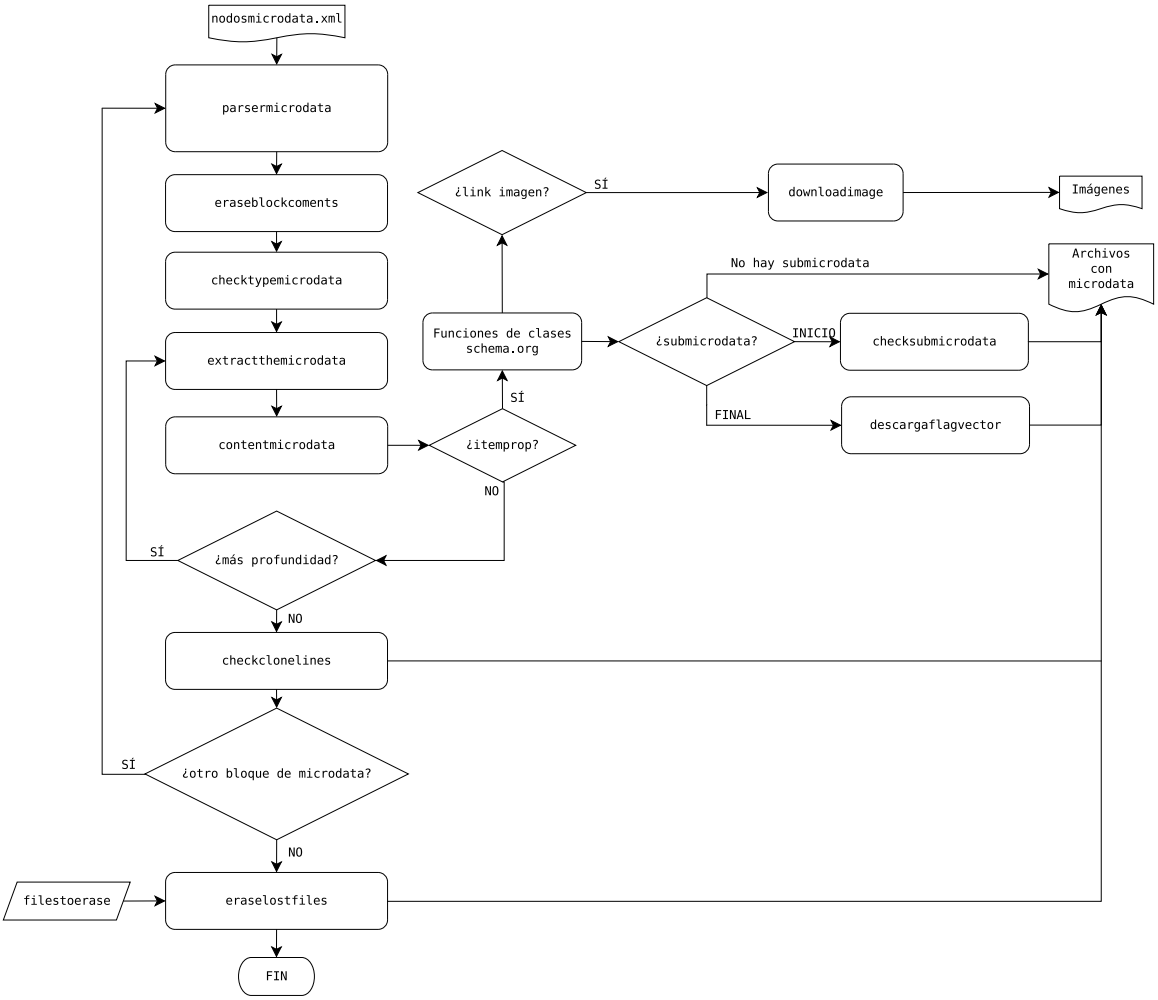

Figura 3.5: Estructura de funcionamiento del cuarto bloque. En él se lleva a cabo el análisis de la microdata encontrada y extraída en la página web descargada, extrayendo dicha información junto a la imagen a la que ésta hace referencia.

A continuación se describirá el funcionamiento de cada una de las partes que permiten el análisis y la extracción de las imágenes y su microdata:

- **parsermicrodata:** Este bloque realiza el control principal del *parser* a la hora de analizar los nodos. El funcionamiento del bloque tiene como objetivo la asignación de nodos al archivo *nodosmicrodata.xml* y la llamada al resto de bloques pasando como argumentos dichos nodos. Estos nodos se asignan mediante el uso de objetos propios de la librería *tinyxml*, y son claves para el análisis y la extracción de la microdata. Dichos objetos son:

  - **TiXmlDocument doc:** En este objeto se almacenará toda la información del documento que se desea analizar (en este caso *nodosmicrodata.xml*) de forma estructurada, permitiendo a la librería *tinyxml* trabajar en cualquier nivel de la estructura del documento, ya sea para solo lectura o también la adición de nuevos nodos en éste.

- **TiXmlNode* nnoderaiz:** Este nodo se asignara a las etiquetas principales de cada bloque de microdata, es decir, sus etiquetas raíz o contenedoras. El bloque asignará el nodo a la primera etiqueta que encuentre en el documento. Tras el análisis de la microdata presente dentro de dicha etiqueta, el nodo se asignará al siguiente nodo hermano (un nodo hermano es un nodo presente en el mismo nivel de la estructura) para el análisis de éste, repitiéndose el proceso hasta que no existan más nodos hermanos y, por tanto, se haya llegado al final del documento.
- **TiXmlNode* nnode:** La misión de este nodo es el análisis de cada etiqueta contenida en el nodo principal *nnoderaiz*. Por ello, este nodo se asignará a los nodos hijos (nodos contenidos dentro de otro nodo) de la etiqueta principal, así como a los nodos hijos de dichos nodos, de forma reiterativa, hasta completar el análisis del bloque de microdata.
- **TiXmlElement* elehtml:** Para poder analizar y leer los atributos e información contenidos dentro de las etiquetas, es necesario usar un objeto tipo elemento. Éste estará siempre vinculado a un nodo, de forma que para analizar el contenido de otra etiqueta será necesario asignarlo de nuevo a un nodo que contenga dicha etiqueta.

Para entender mejor el mecanismo de análisis del bloque, se puede observar la figura 3.6. En ella se muestra un ejemplo de etiquetas con microdata extraídas de varias páginas.

Figura 3.6: Ejemplo de dos bloques con microdata. Puede observarse cómo el nodo *nnoderaiz* se encarga de moverse de bloque en bloque, mientras que *nnode* se mueve por el interior del bloque y el elemento *elehtml* analiza y extrae la información presente en las etiquetas.

Como se puede observar, el nodo *nnoderaiz* se mantiene siempre en las etiquetas superiores, moviéndose a la siguiente etiqueta superior una vez analizada la microdata correspondiente. A su vez *nnode* se mueve por todos los nodos contenidos dentro de *nnoderaiz*, mientras que *elehtml* permite la localización y extracción de la información buscada contenida en éstos.

- **eraseblockcoments:** Este bloque tiene como objetivo la eliminación de los comentarios *HTML* que se puedan encontrar en los nodos con microdata. Esto es debido a que la librería *tinyxml* no soporta estos comentarios, derivando en una lectura inexacta de la estructura.

- **checktypemicrodata:** Es un bloque cuyo objetivo es la lectura de los atributos *itemscope* e *itemype* para la identificación del tipo de microdata que se va a describir. Como ya se ha mencionado anteriormente, existen dos clases de microdata que describen imágenes en las que se centra el programa: *ImageObject* [26] y *Photograph* [27]. El bloque recibe como argumento el nodo *nnoderaiz* que contiene la etiqueta contenedora del bloque de microdata, analizando todos los atributos contenidos en éste. En el caso de encontrar el atributo *itemtype*, se comprobará si pertenece a alguno de los tipos mencionados escribiéndose *MICRODATA OF IMAGEOBJECT* o *MICRODATA OF PHOTOGRAPH* en la cabecera del archivo de salida.

- **extractthemicrodata:** El objetivo de este bloque es controlar el análisis de los nodos contenidos dentro de la etiqueta contenedora de microdata cuyo nodo es *nnoderaiz*. Así, mediante el control del nodo *nnode* y el elemento *elehtml* éste irá llamando al bloque *contentmicrodata* en cada nodo existente para extraer su microdata. Debido a que los bloques de microdata tienen una extensión y profundidad (entendido profundidad como el número de nodos que hay dentro de otros nodos) totalmente variable y poco previsible, es necesario la reiteración del bloque para poder analizar la estructura completa. Dicha reiteración se repetirá el número de veces que sea necesario hasta alcanzar el último nivel de profundidad. La reiteración de este bloque es uno de los factores clave en el rendimiento del programa, siendo éste más lento cuanto mayor sea la profundidad del bloque de microdata a analizar.

- **contenmicrodata:** El objetivo de este bloque es la llamada a las funciones de clases de microdata en el caso de que se detecte el atributo *itemprop*, siendo por tanto un bloque de control.

- **Funciones de clases:** Como puede observarse en la página que define la clase *imageobject* [26] ésta depende de tres clases principales: *thing*, *creativework* y *mediaobject*. A su vez dichas clases dependen de otras tales como *organization, person etc.*, resultando al final una gran variedad de clases posibles que se pueden dar en el análisis de imágenes con microdata. Para analizar todas estas clases, y dado que el objetivo del programa es extraer la información de la microdata, y no la manipulación de ésta, se ha optado por usar una función por clase que

analice los atributos *itemprop* propios de ésta.

Para extraer la información existente en los atributos *itemprop* existen varias posibilidades que dependen del propio atributo, así como de la estructura de la etiqueta:

- **Etiqueta normal o doble:** En este caso la información a extraer es el texto existente dentro de la etiqueta. Es la posibilidad más común y fácil de analizar, como se puede observar en el siguiente ejemplo:

```
<meta itemprop="author">Nikola Tesla</meta>
```

- **Etiqueta única:** Algunas etiquetas en *HTML* no tienen el cierre normal de una etiqueta, sino que cierran la propia etiqueta al final de ésta. En estos casos la información que interesa extraer suele estar almacenada en otro atributo; habitualmente suele ser el atributo *content*, pero en ciertos propiedades de clases como *contentUrl* puede estar almacenado en atributos como *href* o *src*. El mismo ejemplo anterior usando una etiqueta única sería:

```
<meta itemprop="author" content="Nikola Tesla" />
```

- **Llamada a clases anidadas:** Como se ha comentado anteriormente, la microdata permite que existan clases anidadas (en el programa se denominan como submicrodata) dentro de otras clases. De esta forma se puede describir una organización solo por su nombre, o por el contrario aportar más información mediante una clase anidada. En el ejemplo anterior, la propiedad *author* puede describir textualmente el nombre del autor o bien declarar la clase *person* tal como se puede observar a continuación:

```
<meta itemprop="author"  itemscope="" itemtype="http://schema.org/
    Person">
        <div itemprop="name">Nikola Tesla</div>
        <meta itemprop="birthDate" content="10-07-1856" />
        (...)
</meta>
```

Funcionamiento estándar: En general, una función típica de clase seguirá la siguiente estructura: en primer lugar, debido a la existencia de submicrodata o microdata anidada, en ocasiones es necesario almacenar información de microdata para escribirla más tarde. Si al iniciar la función de la clase hay información pendiente de escribir, se escribirá en el archivo antes que el resto. A continuación se comprueba mediante una condición si la propiedad pertenece

o no a dicha clase de microdata. En caso de que no sea así, la función terminará y *contentmicrodata* llamará a la siguiente clase de microdata a analizar. Si dicha propiedad sí pertenece a la clase, se comprueba si la etiqueta donde se sitúa la propiedad es una etiqueta normal, o por el contrario es una etiqueta única, en el caso de ser una etiqueta normal puede extraerse su información de forma genérica. Si por el contrario es una etiqueta única, la información de la microdata podrá encontrarse en distintos atributos. En general hay tres posibilidades:

- **Propiedades tipo *content*:** Serán propiedades cuyo valor está escrito en él código *HTML*, normalmente en formato texto, dentro de la etiqueta *content*.
- **Propiedades a clases anidadas:** Serán propiedades cuyo valor es la declaración de otra clase anidada dentro de la anterior. Éstas llamarán al bloque *checksubmicrodata*, encargado de abrir y declarar una clase anidada, como comentaremos más adelante.
- **Propiedades mixtas y especiales:** Algunas propiedades pueden presentar las dos anteriores opciones, pudiendo tener un valor directo o apuntar a otra clase, así como contener su información en distintos atributos aparte de *content*, por lo que deben tratarse aparte. Es el caso de propiedades como *contentUrl* o *Url*, comunes en la mayoría de las clases de microdata, éstas apuntan a la *URL* de la imagen que se desea descargar, por lo que además llaman a otros bloques del programa como *downloadimage*, para realizar la descarga de la imagen.

Por último, es importante destacar cómo las funciones de clase comprueban continuamente si se está describiendo una clase anidada o no. En el caso de que se encuentre una propiedad que no pertenece a la clase anidada que se estaba describiendo la función llamará al bloque *descargaflagvector* que realizará el cierre de ésta.

- **checksubmicrodata:** Este bloque lee el atributo *itemtype* de la clase anidada encontrada, escribiendo en el archivo de salida el inicio de la clase anidada y comunicándoselo al resto del programa.
- **descargaflagvector:** Este bloque es el encargado de cerrar la descripción de una microdata anidada, escribiendo su cierre en el archivo de salida y comunicándoselo al resto del programa.
- **downloadimage:** El objetivo de este bloque es la descarga de imágenes a una ruta determinada, por lo que su funcionamiento básico es muy similar al del bloque *downloadpage*. A pesar de que éste puede descargar imágenes, se ha optado por crear un bloque específico para la descarga de imágenes que permita una mejor gestión en la descarga de éstas y de sus posibles problemas, como que ya no existan en el servidor, etc.

  Funcionamiento: En el caso de encontrarse una *URL* codificada se descodifica para poder realizar la descarga. A continuación se comprueba si la dirección *URL* es correcta y corresponde

a una imagen. En caso afirmativo, se comprueba si ésta existe ya en el directorio donde se pretende almacenar las imágenes. En caso de que no exista se procede a la descarga de la imagen mediante la librería *libcurl*. Una vez que la imagen ha sido descargada se comprueba que ésta es realmente una imagen. En ocasiones, el servidor no contiene ya la imagen y devuelve a *libcurl* una página *HTML* de error, descargándose ésta como sustituto. Para evitarlo, se lee el archivo binario descargado comprobando que no contenga ninguna cabecera propia de un archivo *HTML*. Al igual que en *downloadpage*, el bloque devolverá el valor *true* en el caso de que la descarga se haya realizado con éxito.

Tras el análisis y la extracción de la microdata e imágenes realizados en este bloque, se habrán obtenido las imágenes y archivos *txt* con la microdata vinculada a éstas, obteniéndose así los archivos de salida del programa.

3.3.5. Bloque 5: Iteración a la siguiente página

El último bloque del funcionamiento básico del programa es un bloque de realimentación cuyo objetivo será la obtención de una nueva *URL* para repetir el proceso. Para ello el bloque leerá el archivo *href.html* donde se había almacenado previamente las direcciones *URL* a las que apuntaba la página descargada, devolviendo la *URL* que se le ha solicitado en su llamada.

Para ello el bloque recibe dos argumentos. El primero corresponde al número de link que se desea extraer, es decir, el número de línea del archivo *href.html* que la función debe extraer. El segundo argumento es el número máximo de links extraídos de la página analizada y que, por lo tanto, deben coincidir con el número de líneas del archivo *href.html*, evitando de esta forma posibles errores. El bloque leerá el archivo mencionado hasta la línea prevista, extrayendo dicha *URL*.

Una vez que hemos obtenido una nueva *URL*, el bloque 1 estará listo para descargar una nueva página y repetir todo el proceso de análisis. Sin embargo, el comportamiento básico del programa descrito en las páginas anteriores, necesita un bloque de control que llame a cada bloque en el momento adecuado, y establezca cual será el comportamiento del *crawler* en la navegación a través de la web, como se comentará a continuación:

3.3.6. Bloque de control: main

El bloque de control *main* tiene los siguientes objetivos:

- **Menú de acceso:** El bloque de control muestra por pantalla y administra la entrada de órdenes para la ejecución del programa. Para ello el programa usa una estructura de menús y submenús de forma arbolada.
- **Comportamiento del crawler:** Un *crawler* o *araña web* tiene muchas opciones para elegir qué página analizará a continuación y, por tanto, cómo se moverá por la web. Dentro del

bloque de control *main* se establece la función *navegar* con este objetivo, así como las funciones *surffreefoto, surfdreamstime, surforistockphoto* encargadas del comportamiento del *crawler* en los casos específicos que se comentarán más adelante.

- **Configuración de parámetros y opciones adicionales:** El programa, como se comentará más adelante, dispone de una serie de parámetros que modifican su comportamiento. Además, existen opciones adicionales como el uso del analizador de datos *Exif*, la administración del archivo de historial, la opción de descarga de imágenes sin microdata a partir de una palabra clave, etc.

A continuación se comentarán los distintos bloques de ejecución o *casos* del programa, siguiendo la estructura presente en su menú cuya que se puede observar en la figura 3.7.

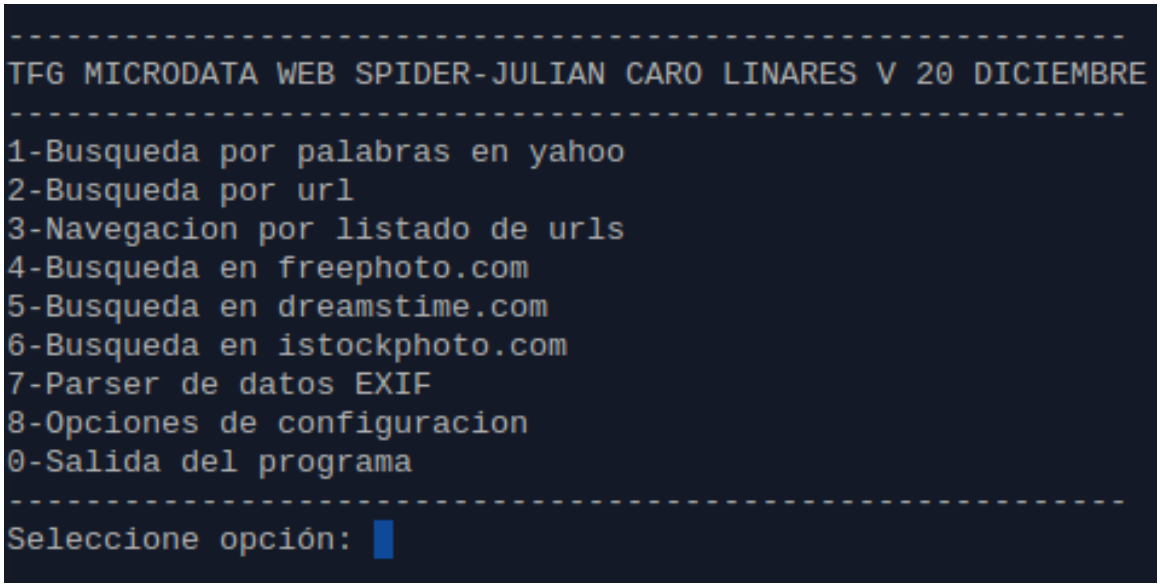

Figura 3.7: Menú principal del programa donde se enumeran las distintas opciones de ejecución.

Caso 1: Búsqueda por palabras en *Yahoo*

Si se selecciona esta opción, el programa comenzará su exploración por la web a partir de los resultados arrojados mediante una búsqueda por palabras clave en el motor de búsqueda *Yahoo* [38]. Para ello este *caso* está dividido en dos partes: obtención de los links de *Yahoo*, y navegación a través de ellos.

En primer lugar se solicitan una serie de datos al usuario, como las palabras claves por las que se desea realizar la búsqueda o el número de páginas de resultados que se desea (siendo 10 resultados por página en el caso de *Yahoo*). Después se le solicitan los parámetros principales que configuraran la navegación en el programa:

- **Número de iteraciones:** Establece el número de iteraciones o *saltos* que debe realizar el programa. A mayor número de iteraciones, mayor posibilidad de analizar más páginas.
- **Porcentaje de enlaces a analizar:** Como se verá con más detalle en la función *navegar*, este parámetro establece el porcentaje de páginas a las que apunta la página descargada actual, que se analizarán y/o iterarán. En ocasiones, frente a ciertas páginas con más de mil enlaces por página, puede resultar útil utilizar un porcentaje bajo para analizar solo una muestra representativa de cada página.
- **Incluir links locales en la búsqueda:** Donde se entiende por links locales enlaces *URL* cuya ruta no es absoluta, sino relativa a la página actual, por lo que necesita una reconstrucción a una *URL* completa para poder ser descargada por el programa. Si se selecciona esta opción se reconstruirán dichos links, por lo que se incrementarán las posibilidades de que el *crawler* analice más páginas del dominio actual frente a dominios externos.

Para la primera parte, la obtención de los links de *Yahoo*, el programa necesita construir la *web request url*, es decir, la *URL* que arrojará los resultados deseados a partir de las palabras clave. En el caso de *Yahoo* la construcción de ésta será la siguiente:

```
1 url="http://search.yahoo.com/search?p="+searchkeywords+"&b="+syahoopage;
```

Donde la variable *url* será la variable que pasaremos al *Bloque 1: Descarga página web (apartado 3.3.1)* para su descarga. Por otra parte, *searchkeywords* contendrá las palabras clave que se desea buscar, donde se han sustituido los espacios por su notación hexadecimal *%20*, y *syahoopage* será el número de páginas de resultados deseado. En el caso de *Yahoo*, esta variable debe incrementarse de diez en diez, por lo que la segunda página de resultados corresponde al valor *11* de *syahoopage*.

Una vez obtenida la *URL* a descargar se llama al bloque 1 *downloadpage*. A continuación se debería llamar al bloque 2 para analizar y extraer los links que interesan. Sin embargo, la mayoría de las páginas de resultados de los motores de búsqueda tienen su código *HTML* escrito en una sola línea, en vez de la habitual disposición de líneas, por lo que el procedimiento que realiza en el bloque 2, analizando línea a línea el *HTML* no es válido para este caso. Por ello se ha desarrollado un bloque específico para la extracción de los resultados de *Yahoo* denominado *processyahoo* que es capaz de leer y extraer la información necesaria a partir de la única línea que contiene todo el código de la página web.

Una vez hemos obtenido los links deseados, se procede a ejecutar la segunda parte: la navegación a partir de éstos. Para ello se llama a la función *navegar*, encargada del comportamiento del *crawler* a través de la web y que se comenta a continuación.

La función navegar: Tiene como objetivo definir el comportamiento estándar que tienen el *crawler* a la hora de moverse por la web.

Dentro de la función existen dos partes de movimiento a otras páginas web bien diferenciadas, cuya suma es lo que se denomina como "una iteración". La primera parte corresponde a la descarga y análisis de todos los links a los que apunta la página web actual, contenidos, como se ha comentado, en el archivo *href.html*. La segunda parte es la selección y descarga de uno de esos links, obteniéndose un nuevo listado de links a analizar. La combinación de la descarga y análisis de la primera página más la función navegar establece las políticas de navegación, comúnmente denominadas como *crawling policies*, que usará el *crawler* para moverse a través de la web. En la figura 3.8 se muestra una iteración de dicho comportamiento.

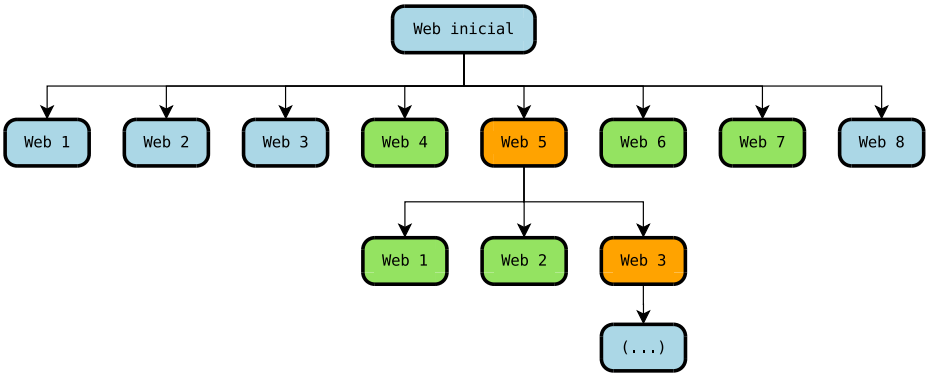

Figura 3.8: Ejemplo de funcionamiento. Las webs marcadas en verde son las que se han analizado según el porcentaje seleccionado, las marcadas en naranja son las que se han seleccionado para realizar una nueva iteración.

Como puede observarse en la figura 3.8, el sistema de navegación es pseudoaleatorio. Una vez que se conoce el porcentaje de enlaces a analizar, así como las iteraciones, se descarga y analizan una porción de los enlaces existentes en la página. Esta porción depende del azar, por lo que si se repite la misma iteración en otro momento se seleccionará otra porción distinta. Una vez se ha analizado la porción de links seleccionada se decide, también de forma aleatoria, a cuál de ellos se va a saltar, descargando de nuevo dicha página y guardando los links a los que apunta. De esta forma se puede repetir el proceso el número de iteraciones que sean necesarias.

La elección de un sistema de navegación pseudoaletorio se basa en el factor de que no conocemos nada de una página hasta que se ha descargado, por lo que se desconoce su posible potencial en cuanto a microdata existente así como otros factores, como links analizados, etc. Sin embargo, debido a la modularidad del programa, se pueden crear en un futuro distintas funciones similares a *navegar* que sigan otras políticas de navegación.

Una vez disponemos de este primer link, la función puede empezar a descargar y analizar los links de la porción seleccionada. Se realizará para cada página todo el proceso de *descarga, procesado, extracción y análisis de microdata* del programa. Es en este momento donde se usa el historial de páginas ya descargadas, creado anteriormente en el bloque *downloadpage*. Se comprueba si el link a descargar ya existe o no en el historial, y por tanto si ya ha sido descargado y analizado. En caso

de que ya exista, se interrumpe el proceso normal de la función, pasando a la repetición del proceso para el siguiente link, lo que evitará que la página vuelva a ser descargada y analizada, aumentando muy significativamente el rendimiento del programa.

Debido a que este uso del historial puede ser, en ocasiones, contraproducente, se han creado diversas opciones que permiten manipularlo, desactivarlo o incluso eliminarlo, como se verá más adelante.

Cabe también destacar que, para aumentar la velocidad del programa, se procesan los códigos *HTML* de cada página sin extraer sus links a *href.html*, ya que se desea analizar solo dicha página y no saltar a uno de ellos, por lo que su extracción es innecesaria.

Una vez que se ha analizado la porción de páginas webs deseada, se procede a elegir de entre esta porción una página web para que sea descargada de nuevo, pero en está ocasión junto con los links a las páginas a los que apunta. Para ello se repite el mismo procedimiento anterior, seleccionando un link mediante el algoritmo pseudoaletorio, descargándolo (*downloadpage*), procesándolo (*processhtml*), extrayendo su microdata (*filterhtml*) y analizándola (*parsermicrodata*).

Estos dos procesos anteriormente comentados se repetirán el número de iteraciones seleccionado, tras lo cual la función terminará. Tras esto, se llama a la función *controlexif* que extraerá los posibles datos *Exif* que puedan existir en las imágenes y cuyo funcionamiento se comentará más adelante.

Caso 2: Búsqueda por url

El funcionamiento de este *caso* es muy similar al *caso 1* de búsqueda por palabras en *Yahoo*, con la diferencia de que en este caso el usuario introduce la *URL* por la que se desea empezar a navegar en vez de palabras clave. Es, por tanto, un caso más general que el anterior, y permite comprobar si existen imágenes con microdata a partir del dominio que deseemos.

Caso 3: Navegación por listado de urls

Este caso permite la navegación e iteraciones a partir de una serie de *URLs* presentes en un archivo. Es, a grandes rasgos, una evolución del *caso 2*, donde se ejecutan las mismas órdenes para cada *URL* del listado presente en el archivo. De esta forma se selecciona una *URL* del listado, se realiza el procedimiento habitual de navegación e iteraciones deseadas y, al terminar, se selecciona la siguiente *URL* del listado repitiéndose el proceso. Es la opción más útil a la hora de realizar grandes barridos a través de la web, ya que permite un alto grado de automatización.

Para que su funcionamiento sea correcto, los archivos a usar deben contener una *URL* por línea, tal como se muestra en la figura 3.9.

Aparte de los parámetros habituales, este *caso* solicita la ruta del archivo a cargar, así como el nombre del directorio donde se desean almacenar los resultados y el número del link del archivo por el que empezar a analizar. Esta última opción es de especial utilidad cuando se desea retomar

Figura 3.9: Ejemplo de listado de *URLs* que se desean analizar. Debe escribirse una sola *URL* por línea.

la búsqueda de un listado de archivos muy extensa que se había comenzado en otra sesión. Al introducir la ruta del archivo a cargar, se crea además un archivo con su mismo nombre, pero con la palabra *results* delante, que contiene cada link analizado junto con una serie de datos, como el número de páginas recorridas o el número de etiquetas de microdata que se han hallando, que resultan útiles para ver de un modo resumido los resultados de una búsqueda extensa. Por lo demás, el funcionamiento de la función será el mismo al del *caso 2*, llamándose a las funciones principales de cada bloque para la descarga, procesado, extracción y análisis de la microdata en cada página web.

Casos 4 y 5: Búsqueda en freefoto y dreamstime

Uno de los casos en los que más se usa el marcado de imágenes con microdata es en los bancos de imágenes. Este tipo de páginas son un repositorio donde encontrar multitud de imágenes sobre cualquier tema y son usadas a diario por multitud de empresas y particulares. Si bien existen casos de acceso gratuito, como el de *freefoto*, la mayoría de bancos de imágenes presentes en Internet son de pago, como en el caso de *dreamstime* por lo que solo es posible descargarse una versión de la imagen con marca de agua. Debido a la importancia que tiene para este tipo de páginas web ofrecer resultados de gran precisión ante las búsquedas de potenciales clientes, estas páginas disponen de una gran base de datos de la que extrapolan, de forma generalmente automática, los datos que se escribirán en la microdata que describirá la imagen. La microdata, así como otros

tipos de marcado por metadatos, son una necesidad fundamental para el modelo de negocio de estos bancos de imágenes, ya que posicionan sus imágenes en los primeros puestos ante una consulta en los principales motores de búsqueda. Por ello la cantidad de microdata ofrecida para describir una imagen en este tipo de páginas es, actualmente, la más completa que se puede encontrar, incluyendo numerosas palabras clave, descripciones extensas, autores y otros campos de gran utilidad.

Por este motivo, en el programa se han desarrollado tres casos especializados en la descarga de imágenes con microdata, a partir de la consulta directa por palabras clave en tres bancos de imágenes distintos. Si bien estas páginas pueden ser analizadas sin problemas con el funcionamiento estándar del programa (por ejemplo usando el *caso 2: Búsqueda por url*), la adaptación de la función *navegar* a estos casos concretos (*surfdreamstime* y *surffreefoto*) presenta la ventaja de poder controlar por completo las iteraciones que se producen dentro del dominio. Al trabajar en un solo dominio, se pasa de un sistema abierto a uno más cerrado donde es posible controlar más parámetros, lo que influye en el número de resultados y en el rendimiento del programa.

Por tanto, estos dos casos presentan la misma estructura básica que el resto de casos, pero usando cada una su propia función simplificada para el movimiento a través de la web qcasosue sustituye a la función estándar *navegar*. La principal diferencia de las funciones ya mencionadas respecto a ésta es su segunda parte, la selección de una página entre las analizadas y su descarga y análisis completo para empezar otra iteración. En estos casos no se selecciona la página web de forma pseudoaleatoria sino que, reconstruyendo la *web request* del dominio, se pasa a la siguiente página de resultados para las palabras clave seleccionadas. De esta forma cada iteración corresponderá a una página de resultados dentro del banco de imágenes, por lo que a más iteraciones, se obtendrán más imágenes con microdata.

Caso 6: Búsqueda en istockphoto

Este caso trabaja también con una página de banco de imágenes. Sin embargo, el planteamiento para su funcionamiento es algo distinto. En este caso se ha buscado la máxima optimización posible del programa para la descarga de imágenes en este dominio, siendo su principal objetivo aumentar la velocidad de análisis y descarga. Para ello, este caso utiliza algunas funciones propias que se comentarán a continuación.

En primer lugar, aunque *istockphoto* originalmente usaba el marcado de las imágenes mediante microdata, actualmente ha implementando al mismo tiempo el uso de *schemaScope*, usado en *XML* para relacionar distintos bloques de información. Este cambio ha propiciado que no se defina correctamente el atributo de microdata *itemtype*, careciendo de la definición del tipo de microdata que se va a describir, por lo que ha sido necesario incorporar un filtro específico en el *Bloque 3: Extracción de código con microdata* (apartado 3.3.3) para la correcta captura de ésta.

También en este caso se ha usado una función propia para sustituir a *navegar*, cuyo nombre es

surforistockphoto. Su funcionamiento es similar a las anteriores salvo por el añadido del uso de la función *istockphotoduplicates*, que permite eliminar enlaces de distinta *url* pero mismo contenido a otros ya analizados. De esta forma el número de enlaces por página de resultados a analizar se reduce a prácticamente la mitad, aumentando de forma significativa el rendimiento de éste.

Por otro lado, este caso no usa las funciones estándar del *Bloque 4: Análisis y extracción de la microdata* (apartado 3.3.4) sino que usa una versión simplificada de éstas. El uso de estas funciones simplificadas permite al programa realizar el análisis de la microdata más rápido, al no tener que comprobar todas las posibles propiedades y clases que podrían existir (*istockphoto* utiliza solo las propiedades *contentUrl, name, description*, la propiedad no estándar *author copyrightHolder* y *keywords*).

Además, a la hora de descargar las imágenes, se ha optado por usar la función *downloadimagesamehost*. El factor que más influye en la velocidad del programa es la apertura de conexiones por parte de la librería *libcurl* para poder realizar la descarga de webs y de imágenes. Debido a que normalmente el *crawler* pasa por diversos dominios de forma frecuente, es necesario que *libcurl* abra y cierre una conexión por vez, lo que repercute en la velocidad del programa. Sin embargo, en este caso, al estar en un sistema más cerrado donde todas las imágenes están alojadas en el mismo dominio, es posible reutilizar la misma conexión. Para ello se almacenarán de forma temporal todas los links a imágenes que se desean descargar, al llamar a la función *downloadimagesamehost* ésta descargará cada link encontrado en el archivo manteniendo la conexión abierta hasta que termine la lectura de éste. La descarga de todas las imágenes de forma consecutiva reutilizando la misma conexión aumenta la velocidad del programa de forma muy significativa.

Caso 7: Parser de datos Exif

Este caso permite la utilización directa del bloque de extracción de datos *Exif*, usado tras la descarga de imágenes en otras partes de programa, para las imágenes contenidas en el directorio que se especifique.

Tal como se ha comentado en el apartado 3.2.1, los datos *Exif* son una serie de metadatos incrustados en el binario de fotografías con formato *JPEG* que contienen información referente al modelo de cámara y parámetros de ésta con los que fueron realizados la fotografía, tales como fecha, ancho y alto, sensibilidad, objetivo usado, apertura del diafragma, velocidad de obturación e, incluso, coordenadas *GPS*. También pueden, en ocasiones, estar presentes en ilustraciones modificadas mediante programas de edición de imágenes. Son, por tanto, un complemento útil a la descripción de una fotografía mediante microdata, aportando más información y complementando a ésta.

En este bloque cabe destacar la función *exifinterprete*. Es una función experimental que, mediante la lectura de los datos *Exif* obtenidos, pretende interpretar dichos datos escribiendo las conclusiones que se pueden extraer de éstos en lenguaje natural. En el campo de la fotografía, se usa a menudo

los datos *Exif* para entender cómo se ha realizado una fotografía, ya que éstos aportan información de cómo se configuró la cámara cuando ésta se realizó. Existen tres factores claves íntimamente relacionados a la hora de realizar una fotografía:

- **ISO o sensibilidad del sensor:** Establece la sensibilidad del sensor respecto a la luz que recibe. A mayor sensibilidad mayor luz se captará pero al mismo tiempo existirán mayores posibilidades de que aparezca ruido electrónico en el sensor, por lo que la calidad de la fotografía será menor (por ejemplo, en bajas condiciones lumínicas, como por la noche).
- **Velocidad de obturación:** Es el tiempo que el obturador de la cámara permite el paso de la luz al sensor. Es decir, el tiempo de exposición o el tiempo que se tarda en realizar la fotografía. Se expresa en milisegundos mediante la notación *1/milisegundos*. Con velocidades altas se puede congelar momentos de un acción (como en la fotografía deportiva) mientras que con bajas se favorece la sensación de movimiento (por ejemplo, el curso del agua en un río).
- **Apertura de diafragma:** Regula la apertura del diafragma del objetivo, permitiendo pasar más o menos luz. Se expresa mediante la notación *f/valor* donde valores bajos indicarán un diafragma más abierto respecto a valores altos. La apertura de diafragma permite, entre otras cosas, controlar la profundidad de campo en una fotografía, o lo que es lo mismo, la cantidad de elementos que aparecerán enfocados en ésta. A valores bajos, obtendremos una profundidad de campo reducida (el caso de un retrato) mientras que a valores altos tendremos una profundidad de campo más grande en la que la mayoría de objetos saldrán enfocados (por ejemplo, una foto de paisaje).

A estos factores se le suman otros parámetros tales como el balance de blancos, el uso del flash, el objetivo utilizado, etc.

Estos factores están fuertemente relacionados con el campo de estudio de la óptica, su manipulación permite a los fotógrafos conseguir los resultados deseados dependiendo del tipo de fotografía que se busca obtener. De esta forma, para realizar un retrato, un fotógrafo profesional tratará de tener una profundidad de campo reducida para enfocar la atención en el sujeto retratado. Para ello el fotógrafo configurará la cámara para tener valores de apertura de diafragma pequeños, así como usará el zoom lo máximo posible, etc.

El lector que esté interesado en profundizar en la teoría fotográfica, puede encontrar explicaciones más extensas del funcionamiento de éstos factores en numerosas publicaciones, tales como *La fotografía paso a paso, Michael Langford* [39] o el *Manual de Técnica Fotográfica, John Hedgecoe* [40].

Es importante destacar que, si bien la teoría fotográfica establece una serie de referencias a la hora de realizar un tipo de fotografía u otra, estos son solo consejos a seguir, no normas de obligado cumplimiento. Como en cualquier arte, el fotógrafo intenta primero dominar las reglas básicas, para

después romperlas a su conveniencia con el objetivo de conseguir transmitir con precisión aquello que desea expresar. De ahí la dificultad que puede tener una función como ésta, cuyo finalidad es intentar interpretar, mediante una serie de datos objetivos como son los datos *Exif*, la intención subjetiva que se buscaba a la hora de realizar la instantánea.

Debido a este motivo, la lógica de la función debe basarse siempre en una hipótesis de mundo abierto (*OWA*) donde solo deben realizarse las suposiciones estrictamente necesarias a partir de los datos disponibles. Para ello, la función usará una serie de sentencias condicionales en la que se compararán diversos valores. Si éstos se cumplen se escribirá una frase en el archivo con la conclusión. Por ejemplo, si se tiene una valor *ISO* de más de *800*, la fotografía habrá sido, muy probablemente, tomada en malas condiciones de iluminación, como en una habitación oscura o por la noche.

Caso 8: Opciones de configuración del programa

El programa dispone de una serie de parámetros configurables, así como funcionalidades extra que se comentarán a continuación:

1. **Activar/Desactivar búsqueda de imágenes por palabra clave:** Esta funcionalidad extra permite la descarga de imágenes sin microdata que contengan una serie de palabras clave. En caso de activarse se buscarán dichas imágenes dentro del *Bloque 2: Análisis del HTML*. Tras la configuración se puede usar cualquiera de las opciones de exploración anteriormente comentadas para realizar la búsqueda.

2. **Opciones de historial de links ya analizados:** Dentro de este menú se pueden encontrar varias opciones que permiten modificar el comportamiento del historial en el programa:

    1) **Activar/Desactivar uso de historial en navegación:** Permite activar o desactivar el historial de navegación. La opción de usar el historial de navegación está por defecto activada, ya que evita que el programa descargue de nuevo webs ya analizadas. Aunque esto normalmente puede suponer una ventaja, en ciertas ocasiones se puede desear no usar el historial usando esta opción para desactivarlo.

    2) **Usar un historial nuevo para esta sesión:** Permite usar un historial de navegación nuevo solo para ésta sesión del programa.

    3) **Usar historial permanente:** Reactiva el uso del historial permanente tras desactivarlo en la opción anterior.

    4) **Importar historial:** Importa el archivo que se especifique al programa para ser usado como historial. El archivo ha de contener una sola *URL* por línea.

    5) **Exportar historial:** Funciona a la inversa que la opción anterior, permitiendo exportar el historial actual a un archivo con el nombre que se especifique.

6) **Eliminar historial:** Elimina el archivo del historial.

3. **Obtener listado de links a partir de url/palabras clave o archivo html:** Esta opción permite generar archivos con listados de *URLs* de forma automática y que se pueden usar en el *caso 3: Navegación por listado de urls* (apartado 3.3.6). Para ello presenta tres opciones:

   1) **Introducir URL:** A partir de una *URL* el programa genera un listado de las *URLs* contenidas en ésta. Para ello descarga la página mediante el bloque *downloadpage* y extrae los links mediante el *Bloque 2: Análisis del HTML*. Como los links se almacenan por defecto en el archivo *href.html* el programa renombra éste con el nombre que se haya especificado.

   2) **Introducir ruta a archivo html:** Su funcionamiento es similar a la opción anterior, solo que en vez de descargar un archivo presente en la web, el bloque 2 analiza un archivo presente en el ordenador. Esta opción es muy útil para extraer listados de links en archivos como los marcadores de un navegador web, que entre otras opciones suelen incluir la opción de exportar éstos a un documento *html*, así como para otros servicios de marcadores online como *delicious*.

   3) **Resultados de busqueda en Yahoo:** Genera un archivo con un listado de los resultados arrojados por *Yahoo* en una búsqueda por palabras clave. Su funcionamiento es similar a los anteriores, solo que usando el mismo sistema de la primera parte del *caso 1: Búsqueda por palabras en Yahoo* (apartado 3.3.6).

4. **Activar/Desactivar análisis de microdata por página:** Activa o desactiva el análisis por página. Si la opción está desactivada se analizará toda la microdata al terminar la exploración por la web, mientras que si está activada, ésta se analizará antes de pasar a la siguiente página.

El programa ha sido diseñado buscando la máxima modularidad posible, de forma que sea fácil localizar en qué bloque puede presentarse un problema determinado, así como ampliar y mejorar cada uno de los bloques pudiendo añadirse nuevas opciones al programa o incluso adaptarse para propósitos distintos al original.

En el siguiente capítulo se mostrará una serie de ensayos en cuyos resultados podrá observarse el comportamiento del programa en distintas situaciones y casos de funcionamiento.

Capítulo 4

Ensayos y resultados

En este capítulo se mostrarán y explicarán una serie de ensayos básicos del funcionamiento del programa. Debido a que el programa funciona a través de la web, siendo ésta un sistema abierto no determinista donde numerosos factores externos que pueden considerarse aleatorios pueden afectar a las mediciones, ha sido necesario realizar cada medición varias veces con el fin de establecer una media y error para cada punto de las gráficas presentadas.

Es importante destacar también la influencia de la velocidad de conexión de la red en la que se han realizado las mediciones. Éstas se han realizado con la red inalámbrica *eduroam* de la Universidad Carlos III de Madrid. Se ha utilizado siempre la misma red con la intención de variar el menor número posible de parámetros, y así generar comparativas menos sesgadas.

El programa *Onewbmicrodata*, dispone de varias opciones a la hora de navegar por la web, desde la introducción de una *URL*, a la búsqueda por palabras clave en *Yahoo*, o el análisis de listados de *URLs*. En cualquiera de estos casos, el programa creará un directorio con el nombre del dominio web analizado o las palabras clave introducidas, donde almacenará los resultados obtenidos. En la figura 4.1 puede observarse un ejemplo típico de los resultados obtenidos tras una exploración del programa.

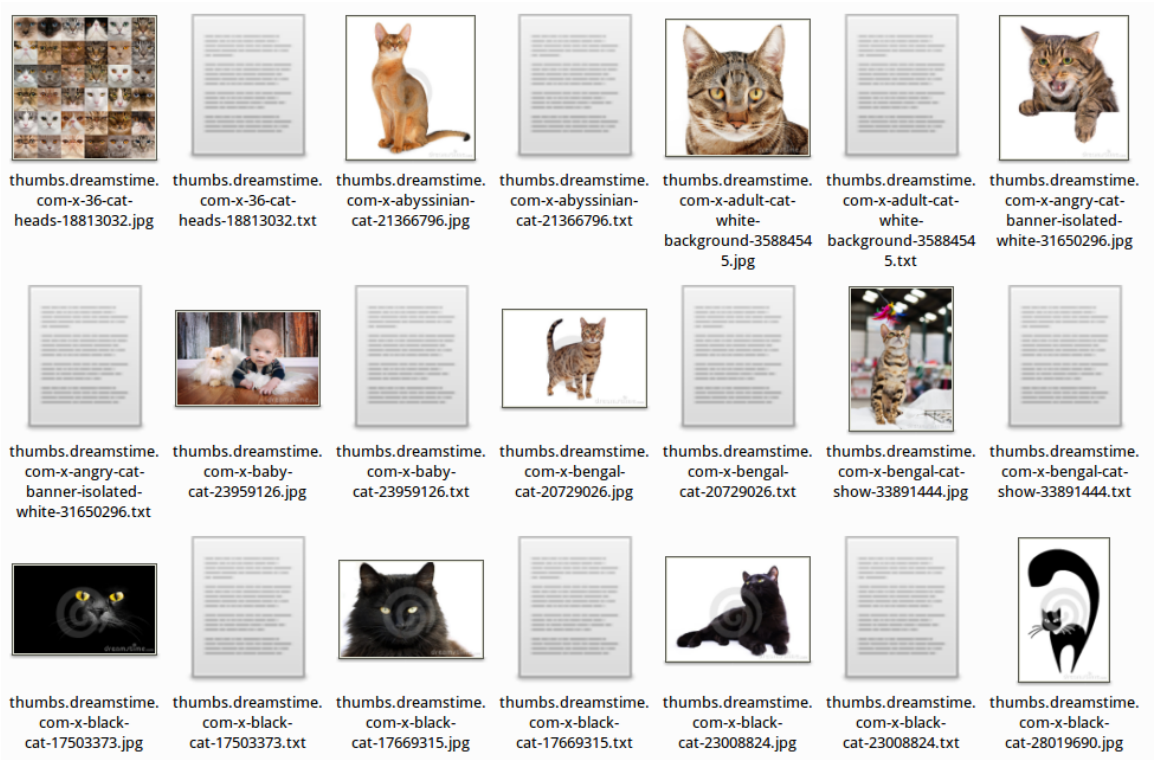

Figura 4.1: Ejemplo de los resultados obtenidos tras una búsqueda con el programa. Cada imagen descargada dispone de un archivo de texto con la información extraída de la microdata que la describía.

Los datos mostrados en las siguientes tablas han sido obtenidos por el propio programa. Para la cuenta de páginas web recorridas en cada medición se ha usado una variable tipo entero que ha hecho las veces de contador. Para el número de resultados se ha observado de forma manual el número de imágenes obtenidas en cada medición. Por último, para hallar el tiempo que ha tardado el programa, se ha usado la función *time* perteneciente a la librería estándar *time.h*. El tiempo se ha empezado a medir a partir de la última introducción de datos requerida por el programa, y se ha parado al finalizar la descarga y análisis de la última página web, por lo que el tiempo mide el tiempo de ejecución real del programa sin tener en cuenta procesos previos como el acceso por menú e introducción de opciones. Dado que el programa en cada medición se mueve en valores del orden de minutos, se ha optado por una resolución de 1 segundo, resolución suficiente para observar las diferencias de tiempo en la repetición de una medición, que suelen variar de varios segundos a varios minutos.

A continuación se presentan los distintos ensayos realizados, mostrando su tabla de resultados y gráficas resultantes, así como una justificación de cada resultado.

4.1. Búsqueda en freefoto.com

En este ensayo se ha usado la opción del programa *Búsqueda en freefoto* (apartado 3.3.6) para obtener imágenes con microdata a partir de la palabra *house*. *Freefoto* es un banco de imágenes donde el usuario puede encontrar y usar imágenes subidas por otros usuarios, en su mayor parte gratuitas. Para realizar el ensayo se han tomado un total de 10 mediciones que corresponden a 10 páginas de resultados de búsqueda en el dominio. Cada medición se ha repetido 5 veces, obteniéndose el resultado final a partir de la media. En la tabla 4.1 se presentan los resultados obtenidos.

MEDICIONES DE LA PALABRA "HOUSE" EN FREEFOTO				
ITERACIONES	TIEMPO (s)	NºPÁGINAS	NºRESULTADOS	DESVIACIÓN TÍPICA TIEMPO
1	290	141	119,6	51,70
2	491	283	240	33,84
3	748	425	360	24,21
4	975,2	567	480	40,38
5	1230,8	709	580	62,49
6	1426,4	851	699	108,68
7	1552,4	993	799	28,29
8	1913	1135	919	151,19
9	2662	1277	1019	234,23
10	2511,6	1419	1136	216,35

Tabla 4.1: Resultados obtenidos al realizar la búsqueda de la palabra *house* en el banco de imágenes *freefoto*.

Si analizamos el número de iteraciones respecto al tiempo, obtenemos la siguiente gráfica (figura 4.2).

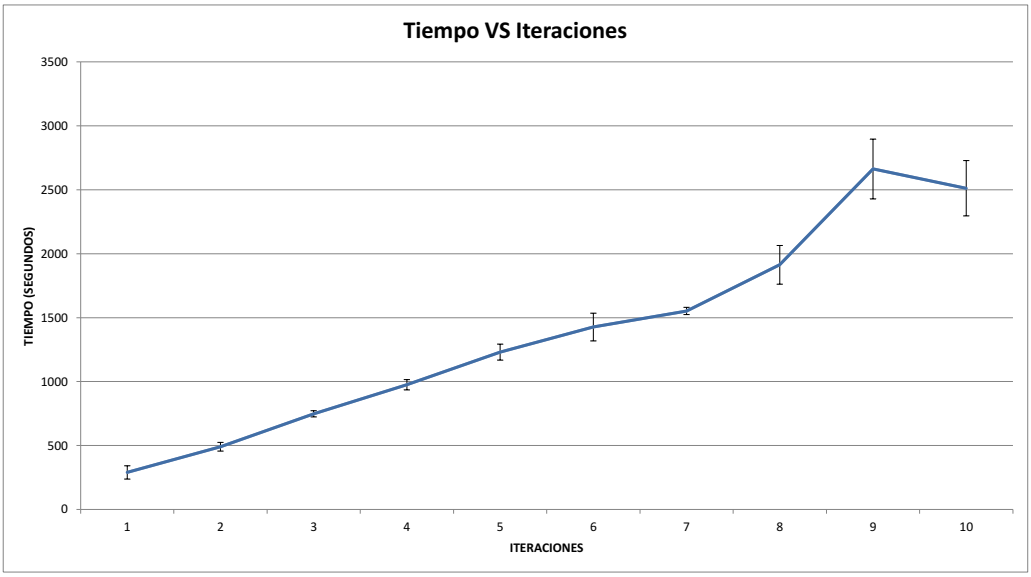

Figura 4.2: Gráfica de tiempo respecto al número de iteraciones (páginas de resultados) realizado. En la gráfica se representan, además, las barras de error del conjunto.

Puede observarse cómo la relación entre el número de iteraciones y el tiempo obtenido es aproximadamente lineal, lo cual es lógico, dado que cada iteración significa una página de resultados más a analizar por el programa. Sin embargo, al llegar a los puntos 9 y 10, dicha linealidad comienza a romperse. Esto se debe a que *freefoto* ofrece para la búsqueda de la palabra *house* un máximo de 1142 resultados. Por lo que al llegar a ese límite, el programa salta a páginas que no corresponden con los resultados de *house* y que tienen menos enlaces y microdata a analizar, estabilizándose, por tanto, el tiempo total que tarda el proceso.

En cuanto a las imágenes descargadas y la información semántica extraída, en la figura 4.3 puede observarse uno de los resultados. Viendo solo la imagen, no se es capaz de extraer información muy detallada sobre lo que representa, solo se puede concluir la presencia de una casa en mal estado.

Figura 4.3: Imagen obtenida a partir de la búsqueda de la palabra *house* en el banco de imágenes *freefoto*.

Sin embargo si tenemos en cuenta los datos obtenidos a partir de la microdata:

```
1  MICRODATA OF PHOTOGRAPH: http://www.freefoto.com/preview/13-04-44/Derelict-
       House HOST: http://www.freefoto.com
2
3  ITEMPROP: name=Derelict House
4  ITEMPROP: description=Pictures of a derelict house
5  ITEMPROP: image=http://s3.freefoto.com/images/13/04/13_04_44_thumb.jpg
6
7  SUBMICRODATA IMAGEOBJECT
8  ITEMPROP: thumbnail=http://s3.freefoto.com/images/13/04/13_04_44_thumb.jpg
9  ITEMPROP: name=/images/13/04/13_04_44---Derelict-House_web.jpg
10 ITEMPROP: representativeOfPage=true
```

```
ITEMPROP: contenturl=http://www.freefoto.com/images/13/04/13_04_44---
    Derelict-House_web.jpg
FIN SUBMICRODATA TIPO->imageobject

ITEMPROP: description=Pictures of a derelict house
SUBMICRODATA PERSON
ITEMPROP: url=http://www.ianbritton.co.uk
ITEMPROP: name=Ian Britton
FIN SUBMICRODATA TIPO->person

ITEMPROP: publisher=Freefoto.com
ITEMPROP: isFamilyFriendly=true
ITEMPROP: keywords=united kingdom , 13-04-0 , 13-00-0 , newcastle upon tyne
    , tyneside , house , demolition , industry , derelict house , falling
    down FreeFoto.com, Free Pictures, Stock Photography, Royalty Free Images
    , Picture, Image, free-use images
ITEMPROP: datePublished=Jan 28, 2008 10:20:54 AM

SUBMICRODATA PLACE
ITEMPROP: contentLocation=Location:

SUBMICRODATA GEOCOORDINATES
ITEMPROP: latitude=54.9663
ITEMPROP: latitude=-1.5653
FIN SUBMICRODATA TIPO->geocoordinates

FIN SUBMICRODATA TIPO->place
```

En este caso, si observamos la información extraída se concluye en primer lugar que la imagen es una fotografía. En la cabecera del archivo aparece el tipo de clase descrito (*imageobject* o *photograph*) en este caso *photograph*, también aparece la *URL* a la web donde ha sido hallada la imagen y la *URL* general de su dominio (*HOST*). Tras la cabecera aparecen las propiedades descritas de la clase, como el nombre de ésta, así como una descripción que indica que es una casa abandonada, además de la *URL* de la miniatura de la imagen. A continuación aparece submicrodata, o microdata anidada. En este caso se describe características de un objeto tipo imagen dentro de la fotografía, y se obtiene información como la *URL* de la miniatura de ésta, su nombre, si la imagen representa bien el contenido de la página de donde ha sido extraída (*representativeofpage*) así como la *URL* a la imagen completa. Una vez que se ha descrito la submicrodata *imageobject*, la página vuelve

a presentar una descripción de la imagen. A continuación se presenta otra microdata anidada. En este caso es una clase *persona* en la que se presenta información sobre el autor de la foto, como su nombre y página web (Ian Britton es uno de los principales fotógrafos que alimentan este banco de imágenes). Tras describir al autor se aporta más información clave de la fotografía, como su editor, si es apta para todos los públicos (*isFamilyFriendly*), palabras claves que la describen, entre las que se encuentra la palabra *house*, que hemos usado en la búsqueda para obtener esta imagen. A continuación, como submicrodata, se presenta el lugar donde ha sido realizado la fotografía, el campo *contentLocation* es usado en este caso de forma incompleta, conteniendo solo la palabra *Location:*, cuando debería contener una descripción del lugar. Dentro de la submicrodata tipo *place* nos encontramos la microdata anidada *geocoordinates*, que contiene las coordenadas de latitud y longitud donde ha sido realizada la fotografía.

Con esta información, un programa puede pasar de conocer solo que está ante un archivo tipo imagen, sin conocer su contenido, a conocer datos tan diversos como su título, descripción, palabras clave que definen la fotografía, persona que la realizó, e incluso, las coordenadas donde se realizó ésta. Las posibilidades de que un programa pueda acceder y comprender éste tipo de información asociado a una imagen pueden llegar a ser muy numerosas y diversas.

4.2. Búsqueda en dreamstime.com

En este ensayo se ha usado la opción del programa *Búsqueda en dreamstime* (apartado 3.3.6) para obtener imágenes con microdata a partir de la búsqueda con la palabra *cat*. Al igual que *freefoto dreamstime* es un banco de imágenes, solo que en este caso sus imágenes no son gratuitas. Su negocio es la venta de imágenes a empresas y particulares. Debido a este motivo, el programa solo puede acceder a la descarga de las versiones de prueba de las imágenes, que contienen una marca de agua para evitar su uso comercial. Para realizar el ensayo se ha tomado de nuevo un total de 10 mediciones, correspondientes a 10 páginas de resultados de la búsqueda realizada. Cada medición se ha repetido 5 veces, obteniéndose la medición final a partir de su media. En la tabla 4.2 se presentan los resultados que se han obtenido.

MEDICIONES DE LA PALABRA "CAT" EN DREAMSTIME				
ITERACIONES	TIEMPO (s)	NºPÁGINAS	NºRESULTADOS	DESVIACIÓN TÍPICA TIEMPO
1	278	206	80	184,13
2	692,4	413	157,4	86,56
3	795	620	237,6	20,55
4	940	827	314,6	46,73
5	1365,8	1034	391	42,61
6	1623,8	1241	469,6	163,15
7	1951	1448	539,8	67,97
8	2281,2	1655	630,2	184,13
9	2520,8	1862	700	227,19
10	2682,6	2069	799	455,32

Tabla 4.2: Resultados obtenidos al realizar la búsqueda de la palabra *cat* en el banco de imágenes *dreamstime*.

Siendo la relación del número de iteraciones respecto al tiempo la observada en la gráfica de la figura 4.2.

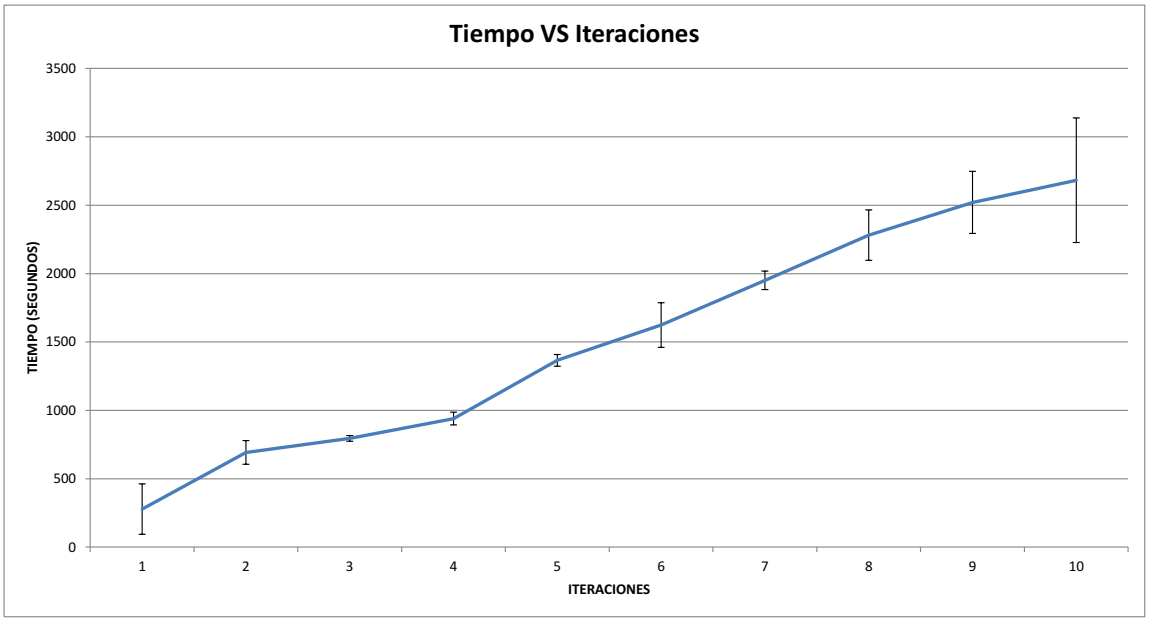

Figura 4.4: Gráfica de tiempo respecto al número de iteraciones (páginas de resultados) realizado en *dreamstime*. En la gráfica se representan, además, las barras de error del conjunto.

Como puede observarse, en el caso de *dreamstime* la relación entre el número de iteraciones y tiempo es completamente lineal, por lo que a mayor número de iteraciones, se descargarán mayor número imágenes con microdata.

A continuación la información con microdata extraída de uno de los resultados obtenidos (figura 4.5).

Figura 4.5: Imagen obtenida a partir de la búsqueda de la palabra *cat* en el banco de imágenes *dramstime*.

La información extraída de la microdata es la siguiente:

```
1  MICRODATA OF IMAGEOBJECT: http://www.dreamstime.com/stock-photography-cat-
       looking-up-image13042572 HOST: http://www.dreamstime.com
2
3  ITEMPROP: image=http://thumbs.dreamstime.com/x/cat-looking-up-13042572.jpg
4
5  SUBMICRODATA AGGREGATERATING
6  ITEMPROP: ratingValue=5
7  ITEMPROP: worstRating=0.5
8  ITEMPROP: bestRating=5
9  ITEMPROP: ratingCount=2
10 FIN SUBMICRODATA TIPO->aggregaterating
11
12 ITEMPROP: description=Domestic cat looking up (B/W).
13 ITEMPROP: name=Stock Photography:
14 ITEMPROP: author=Ambience
15 ITEMPROP: keywords=soft
16 ITEMPROP: keywords=look
17 ITEMPROP: keywords=gray
18 ITEMPROP: keywords=curiosity
19 ITEMPROP: keywords=tabby
20 ITEMPROP: keywords=face
21 ITEMPROP: keywords=feline
22 ITEMPROP: keywords=nap
23 ITEMPROP: keywords=mammal
24 ITEMPROP: keywords=black
```

```
25 ITEMPROP: keywords=domestic
26 ITEMPROP: keywords=pet
27 ITEMPROP: keywords=hair
28 ITEMPROP: keywords=whiskers
29 ITEMPROP: keywords=text
30 ITEMPROP: keywords=love
31 ITEMPROP: keywords=closeup
32 ITEMPROP: keywords=kitty
33 ITEMPROP: keywords=close
34 ITEMPROP: keywords=animal
35 ITEMPROP: keywords=fear
36 ITEMPROP: keywords=companion
37 ITEMPROP: keywords=watching
38 ITEMPROP: keywords=sweet
39 ITEMPROP: keywords=cat
40 ITEMPROP: keywords=white
41 ITEMPROP: keywords=background
42 ITEMPROP: keywords=eyes
```

Como puede observarse, se describe una clase tipo *imageobject*. En ella está anidada la clase *aggregaterating* que clasifica la imagen según los votos que ha recibido por los usuarios. Esta clase es una de las más comunes en el uso de la microdata en la web, siendo frecuentemente usada para la descripción de elementos multimedia (*mediaobject*) como vídeos, imágenes, canciones etc., o en la descripción de productos (*product*) y negocios, tales como restaurantes (*foodestablishment*). Dentro de ella, se describe el valor actual (5 estrellas en este caso) el peor valor votado (0.5 estrellas) el más alto y el número de votaciones (*ratingcount*).

A continuación se realiza una descripción de la imagen, dándose además su nombre y el autor de ésta. En este caso no se recurre a una clase *person* para establecer el autor, como en el caso visto en *freefoto*, sino que se aporta solo el nombre de éste. A continuación aparecen una serie de palabras clave o *keywords* que describen la imagen. Cabe destacar la calidad y el elevado número de éstas. Los bancos de imágenes como *dreamstime*, cuyo modelo de negocio es la venta de imágenes, ganan dinero cuando el cliente encuentra justo la imagen que estaba buscando, comprándola. Ofrecer un motor de búsqueda interno lo más completo y potente posible es fundamental, de ahí que este tipo de páginas dispongan de bases de datos con gran volumen de información para cada imagen. El uso de parte de esta información como microdata, permite al banco de imágenes que los motores de búsqueda coloquen sus imágenes en las primeras posiciones ante la búsqueda de un usuario, siendo un mecanismo de gran importancia para la captación de clientes .

Actualmente, los bancos de imágenes son los que más utilizan las imágenes con microdata, si bien cada vez más sitios web, tales como periódicos digitales, están aportando esta información en sus noticias, todavía con poco nivel de detalle. Es previsible que en un futuro, junto con la expansión de la web semántica, más páginas web usen este tipo de marcado, enriqueciendo la información de las imágenes presentes en la red.

4.3. Búsqueda en istockphoto.com

En este ensayo se ha usado la opción del programa *Búsqueda en istockphoto* (apartado 3.3.6) para obtener imágenes con microdata a partir de la palabra clave *cat*. *Istockphoto*, al igual que *dreamstime*, es un banco de imágenes que gana dinero con la venta de éstas, por lo que el programa solo puede descargar la versión con marca de agua de las imágenes. Para realizar este ensayo se han tomado 10 mediciones correspondientes a 10 páginas de resultados, cada medición se ha repetido 5 veces, obteniéndose la medición final a partir de su media. En la tabla 4.3 se presentan los resultados obtenidos.

MEDICIONES DE LA PALABRA "CAT" EN ISTOCKPHOTO				
ITERACIONES	TIEMPO (s)	NºPÁGINAS	NºRESULTADOS	DESVIACIÓN TÍPICA TIEMPO
1	364,8	107	100	137,97
2	506	213	200	17,49
3	824,6	319	300	57,60
4	1509	425	400	553,25
5	1318,4	531	500	74,65
6	1817,2	637	600	141,30
7	2280	743	700	208,32
8	2617,2	849	798,8	362,39
9	3575,4	955	900	507,69
10	4036,4	1035,5	1000	642,42

Tabla 4.3: Resultados obtenidos al realizar la búsqueda de la palabra *cat* en el banco de imágenes *istockphoto*.

El tiempo que tarda respecto al número de iteraciones es el que se observa en la figura 4.6.

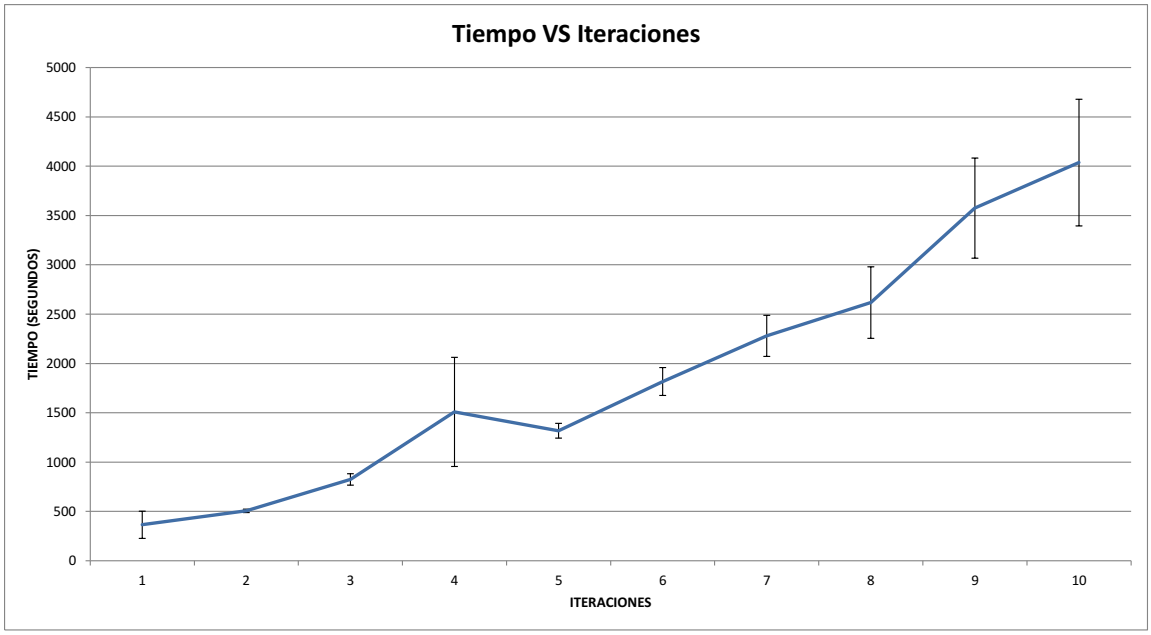

Figura 4.6: Gráfica de tiempo respecto al número de iteraciones (páginas de resultados) realizado en *istockphoto*. En la gráfica se representan, además, las barras de error del conjunto.

Puede observarse como, al igual que los casos anteriores de *freefoto* y *dreamstime*, el número de iteraciones o profundidad guarda una relación lineal respecto al tiempo, aun siendo más inestable que los casos anteriores.

A continuación, en figura 4.7 se muestra uno los resultados obtenidos:

Figura 4.7: Imagen obtenida a partir de la búsqueda de la palabra *cat* en el banco de imágenes *istockphoto*.

Siendo la microdata extraída:

```
MICRODATA OF IMAGEOBJECT: HOST: http://www.istockphoto.com

ITEMPROP: contentUrl=http://i.istockimg.com/file_thumbview_approve
    /16156731/2/stock-photo-16156731-cute-kitten-is-climbing-on-the-rope.jpg
ITEMPROP: name=cute kitten is climbing on the rope - Stock Image
ITEMPROP: description=a cute kitten is climbing on the rope. isolated on a
    white background
ITEMPROP: author copyrightHolder=s-dmit

ITEMPROP: keywords=Domestic Cat
ITEMPROP: keywords=Kitten
ITEMPROP: keywords=Hanging
ITEMPROP: keywords=Cute
ITEMPROP: keywords=Rescue
ITEMPROP: keywords=Rope
ITEMPROP: keywords=Animal
ITEMPROP: keywords=Pets
ITEMPROP: keywords=Gripping
ITEMPROP: keywords=Paw
ITEMPROP: keywords=Young Animal
ITEMPROP: keywords=Isolated
ITEMPROP: keywords=Close-up
ITEMPROP: keywords=Domestic Animals
ITEMPROP: keywords=Studio Shot
ITEMPROP: keywords=Isolated On White
ITEMPROP: keywords=One Animal
ITEMPROP: keywords=Animals And Pets
ITEMPROP: keywords=Scottish Fold
ITEMPROP: keywords=Feline
ITEMPROP: keywords=Purebred Cat
ITEMPROP: keywords=Tabby
```

Como puede observarse, la información obtenida es muy similar a los casos anteriores: la *URL* que contiene la imagen a su nombre, descripción, autor de la imagen y un gran número de palabras clave de gran utilidad a la hora de describir la misma.

Actualmente los bancos de imágenes son las páginas web que ofrecen mayor número, y calidad en su descripción semántica con microdata, de imágenes en la web. Si bien éstas suelen valer dinero

por lo que no son facilmente accesibles, es de esperar que en un futuro próximo servicios sociales de imágenes, como *fliker*, *pinterest* y otros, utilicen microdata para la descripción detallada de sus imágenes, lo que aumentará de forma considerable la presencia de imágenes con información semántica en la web.

4.4. Datos Exif

Aunque gran parte de las fotografías subidas a la web carecen de datos *Exif*, ya sea por su eliminación voluntaria o por su perdida durante su conversión en la edición de un programa de tratamiento de imágenes, muchas otras contienen esta información de gran utilidad para aquel que desee conocer como se ha realizado una fotografía.

A continuación se muestra un ejemplo de fotografía con datos *Exif* hallada en la web.La imagen ha sido encontrada por el programa en el dominio *alanpeto.com* y pertenece al fotógrafo Chi King. Retrata una estatua gigante de Buda en la región de Sichuan, China. Tal como se puede observar en la figura 4.8.

Figura 4.8: Imagen obtenida en el dominio *alanpeto.com*, Chin King, 2007. En ella se observa la estatua gigante de Buda en la provincia de Sichuan, China. La imagen dispone tanto de microdata como de datos Exif.

Archivo con la información extraída:

```
MICRODATA OF IMAGEOBJECT: http://www.alanpeto.com/buddhism/understanding-mahayana-theravada/ HOST: http://www.alanpeto.com
ITEMPROP: image=http://www.alanpeto.com/wp-content/uploads/2013/06/2165330073_a4aa51134d_o.jpg

EXIF METADATA
Camera make       : Canon
Camera model      : Canon EOS 5D
Software          : Adobe Photoshop CS2 Macintosh
Bits per sample   : 0
Image width       : 4076
Image height      : 2912
Image description :
Image orientation : 1
Image copyright   :
Image date/time   : 2007:12:31 21:42:03
Original date/time: 2007:12:26 13:25:10
Digitize date/time: 2007:12:26 13:25:10
Subsecond time    :
Exposure time     : 1/199
F-stop            : f/4
ISO speed         : 640
Subject distance  : 0
Exposure bias     : 0 EV
Flash used?       : 1
Metering mode     : 5
Lens focal length : 32 mm
35mm focal length : 0 mm
GPS Latitude      : 0 deg (0 deg, 0 min, 0 sec)
GPS Altitude      : 0
GPS Longitude     : 0 deg (0 deg, 0 min, 0 sec)
GPS Altitude      : 0

INTERPRETACION DATOS EXIF
Imagen modificada por software Adobe Photoshop CS2 Macintosh tomada originalmente en horizontal
```

Como se puede observar, la imagen no contiene apenas información de microdata, solo el campo *image* que aporta la *URL* de la imagen. Sin embargo sí que dispone de datos *Exif*. Estos datos describen los distintos aspectos técnicos y parámetros configurados en el momento en el que se realizó la fotografía. Destacan sobre todo, tal como se describió en el apartado 3.3.6, el tiempo de exposición, la apertura de diafragma (*F-stop*) y la velocidad *ISO* o sensibilidad del sensor. Estos tres aspectos son de vital importancia a la hora de entender cómo ha sido realizada una fotografía y, por tanto, cómo puede ser reproducida de nuevo. Por último se puede observar una interpretación básica basada en los datos *Exif* donde se concluye que la imagen ha sido retocada por software y fue tomada con la cámara en horizontal.

Los metadatos *Exif* son en muchas ocasiones un gran complemento a la información obtenida a través de la microdata. Aunque puede parecer en principio que los datos *Exif* son demasiado técnicos para ser de utilidad, si se relacionan con otros datos presentes en la microdata pueden llegarse a realizar nuevas inferencias que aporten más contexto a la imagen.

4.5. Un ejemplo de búsqueda múltiple

En el siguiente ensayo se busca comprobar el correcto funcionamiento del programa para búsquedas largas, a veces de varias horas, que conlleven un gran volumen de información a analizar. Para ello se ha usado la opción del programa *caso 3: Navegación por listado de URLs* para analizar un conjunto de 100 dominios que supuestamente tienen imágenes con microdata. Este listado ha sido elaborado a partir del buscador de información semántica *Sindice* [41]. En total se ha realizado una búsqueda de una profundidad o número de iteraciones de 10 para cada una de las 100 páginas web presentes en el listado. Al final de cada dominio el programa ha registrado información como el número de páginas recorridas y el tiempo realizado, así como el número de páginas recorridas y tiempo total. A continuación se mostrarán distintas gráficas que muestran el comportamiento del programa durante el análisis. En la figura 4.9, se muestra una gráfica con el número de páginas analizadas en cada uno de los 100 dominios analizados.

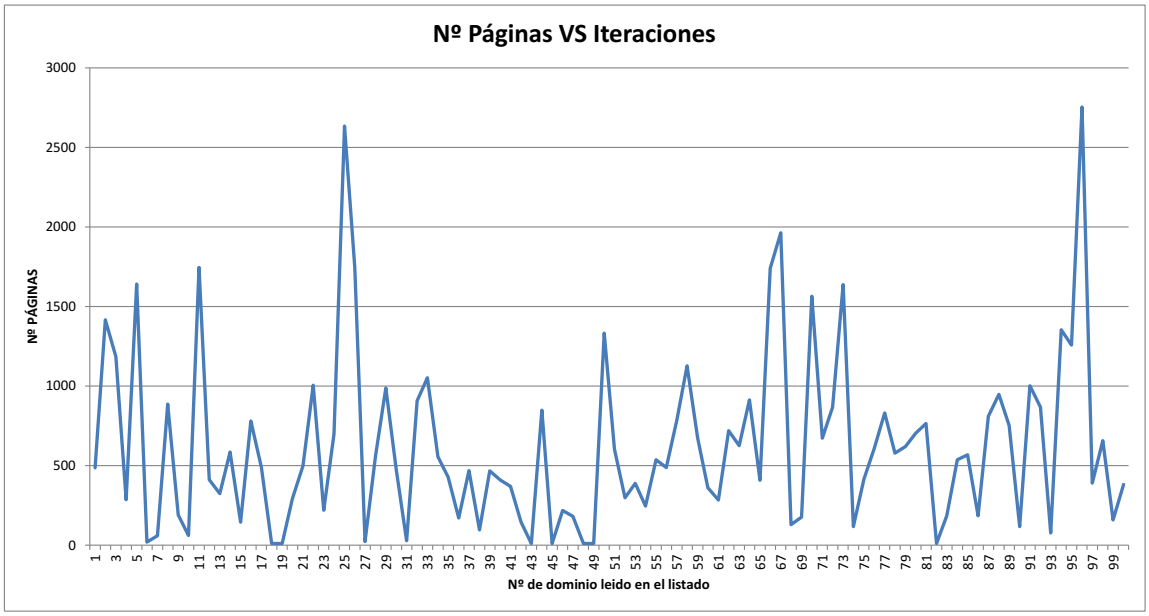

Figura 4.9: Gráfica del número de páginas analizadas en cada uno de los 100 listados del dominio.

Puede observarse cómo la cantidad de páginas analizadas varía dependiendo de la estructura de cada dominio. Los puntos más bajos corresponden con los dominios que el programa no puede analizar. Si a continuación exponemos estos datos en una misma gráfica junto con el tiempo que se tardó en analizar cada dominio (figura 4.10), puede llegarse a la conclusión de que existe una relación directa entre el número de páginas y el tiempo usado durante el análisis, donde a mayor número de páginas, más tiempo se tarda. Cabe destacar el punto de la gráfica número 81, el cual tiene un valor de tiempo anormalmente alto comparado con el resto de dominios. Este tiempo no puede deberse a una pérdida de conexión ya que entonces sucedería que el punto tendría un valor de páginas recorridas y tiempo mínimo.

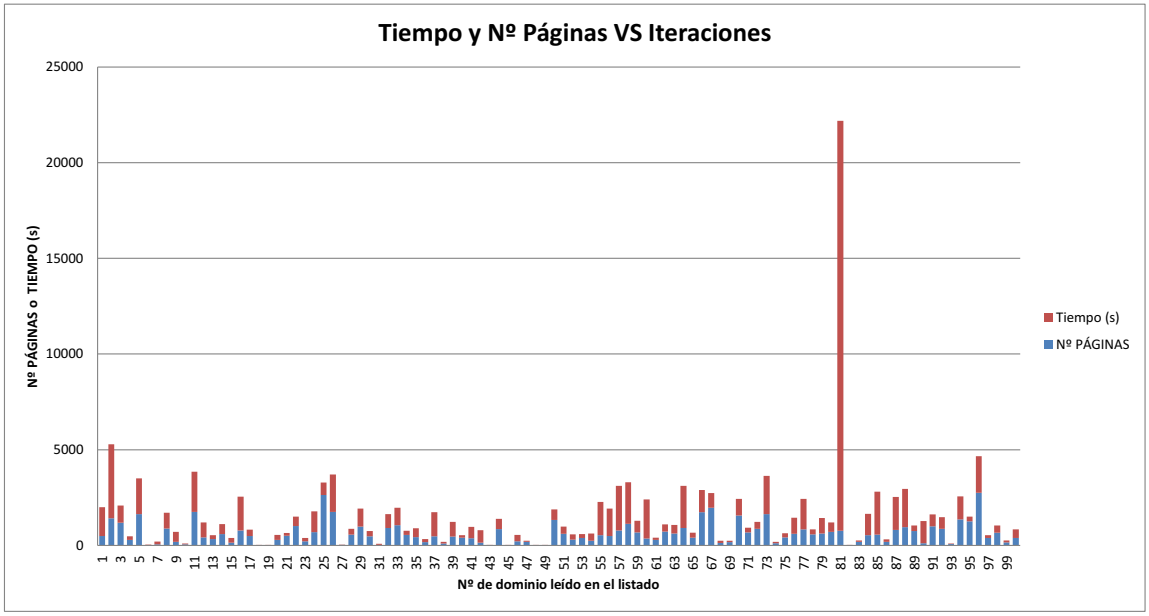

Figura 4.10: Gráfica del tiempo que se ha tardado en analizar cada uno de los 100 listados del dominio junto con el número de páginas recorridas. En general puede observarse una relación directa entre el número de páginas y el tiempo realizado.

Tras repetir la medición varias veces, obteniendo valores similares, se descartó que fuera un error en la medición. La justificación de por qué dicho punto tarda tanto puede encontrase si se observa la carpeta que contiene sus resultados obtenidos. El dominio, llamado *instaweb.me*, es una página especializada en mostrar fotos de la red social *Instagram*. Durante su análisis se encontraron, analizaron y extrayeron 15.877 imágenes, lo que justifica el elevado tiempo en el que el programa permaneció en él.

En total, el programa durante este ensayo recorrió más de 62.432 páginas en un tiempo total de 90.142 segundos, o lo que es lo mismo, más de 25 horas de funcionamiento.

Capítulo 5

Conclusiones

En este capítulo se expondrán, por una parte, las conclusiones del programa desarrollado, enumerando los objetivos alcanzados y dificultades encontradas. Por otro lado, se hablará sobre los posibles desarrollos futuros del programa realizado, así como de la evolución e influencia que tendrá la web semántica en la web del futuro.

5.1. Conclusiones sobre el programa desarrollado

Tras el desarrollo del proyecto, a continuación se enumeraran los objetivos clave que se han alcanzado durante el transcurso del mismo:

1. **Descarga de información:** El programa es capaz de descargar distintos tipos de archivos presentes en la web, descargando las páginas e imágenes deseadas y evitando la descarga del resto de archivos. El uso de la librería *libcurl* permite tener un gran control en las descargas realizadas, así como evitar la mayoría de los principales problemas a los que se enfrenta un programa de esta naturaleza al navegar por la web, tales como redireccionamientos, *URLs* codificadas, páginas vacías, servidores que no responden, etc.

2. **Lectura, modificación y análisis de archivos:** El programa es capaz de leer todo el código *HTML* de las páginas web descargadas mediante la lectura línea por línea del archivo. Además, es capaz de localizar las palabras claves que le interesan, eliminando, sustituyendo o modificando las líneas del archivo según sus necesidades. El método de búsqueda de palabras clave en el archivo, a partir de la generación de un listado de éstas, permite modificar y ampliar de una forma sencilla la lectura y edición del código *HTML*, adaptando éste para la extracción de la microdata u otras futuras aplicaciones.

3. **Obtención y edición de *URLs*:** El programa puede localizar y extraer las distintas *URLs* presentes en una web con el objetivo de descargarlas y analizarlas *a posteriori*. A su vez, el

programa es capaz de diferenciar entre *URLs* de ruta completa y *URLs* de ruta parcial, siendo capaz de transformar estas últimas a rutas completas que puedan ser descargadas.

4. **Localización de la microdata:** El programa detecta si el código de una página web contiene información con microdata, adaptando dicho código para su extracción. Si bien el programa ha sido desarrollado para la extracción de imágenes con microdata, el programa puede ser ampliado fácilmente a la extracción de otro tipos de microdata presentes en la web.

5. **Extracción del código con microdata:** El programa identifica los bloques con microdata presentes en la webs analizadas, extrayendo éstos a otro archivo, lo cual facilita su análisis, así como la detección y corrección de errores. Al analizar solo las etiquetas contenidas en los bloques de microdata, y no todas las etiquetas presentes en la web, se mejora de forma drástica el rendimiento del programa.

6. **Identificación de la microdata:** El programa es capaz de identificar qué bloques de microdata contienen las clases buscadas, en este caso las clases *imageobject* y *photograph* orientadas a la descripción de imágenes, descartando el resto y extrayendo solo las que son de interés. Dicha identificación resulta fácilmente ampliable a otros tipos de microdata que se puedan desear extraer en un futuro.

7. **Análisis y extracción:** El programa analiza los bloques de microdata hallados, extrayendo su información a un documento *txt* y descargando las imágenes descritas.

8. **Descripción de clases anidadas:** El programa además, es capaz de describir las clases dependientes de las clases principales, que pueden aparecer de forma anidada dentro de éstas, identificándolas como un bloque propio dentro del bloque general.

9. **Extracción de datos *Exif*:** El programa puede extraer los datos *Exif* que estén presentes en las fotografías descargadas, añadiendo dicha información a la extraída a partir de la microdata.

10. **Navegación por la web:** El programa es capaz de navegar por la web mediante diferentes medios, desde la introducción de una *URL* a la búsqueda a partir de palabras clave en *Yahoo* o la búsqueda a partir de un listado de *URLs*.

11. **Bancos de imágenes:** El programa es capaz de descargar las imágenes con microdata, a partir de un conjunto de palabras clave, de tres de los principales bancos de imágenes existentes en la web, *freefoto, dreamstime* e *istockphoto*.

12. **Parametrización del programa:** El programa permite modificar los parámetros que influyen en la navegación, desde el número de iteraciones o profundidad de la búsqueda, al uso de *URLs* de ruta parcial y otras opciones. Destaca la posibilidad de generar un historial para evitar la descarga de páginas ya analizadas, mejorando su rendimiento.

13. **Sistema de archivos:** El programa genera una serie de archivos para realizar sus funciones, desde el código original de la página descargada, a este mismo código editado, así como un listado de las *URLs* encontradas y pendientes de visitar de la página web descargada, además de un archivo donde se extraen todos los bloques de microdata encontrados y analizados durante esa sesión del programa, lo que permite, en caso de la interrupción del programa, recuperar y descargar las imágenes con microdata que estaban pendientes de extraer. Este sistema de archivos, donde cada bloque principal del programa genera un archivo usado que es usado por el siguiente, permite un análisis sencillo y efectivo ante la presencia de problemas o comportamientos inesperados en alguna página web. A su vez permite una sencilla adaptación y creación futura de posibles nuevas funcionalidades.

14. **Modularidad:** El programa, formado por cinco bloques de funcionamiento diferenciados por su función, ha sido creado con la idea de tener la mayor modularidad posible, pudiéndose crear *crawlers* con otros objetivos distintos al original a partir de éste. Dado que la web es un sistema abierto altamente cambiante, es importante que un programa tipo *araña web* o *crawler* como el presente pueda adaptarse a los nuevos cambios o tareas que se requieran para él.

15. **Funciones adicionales:** El programa es además capaz de realizar funciones adicionales tales como la descarga de imágenes sin microdata a partir de palabras clave, la extracción de los datos *Exif* existentes en las imágenes del directorio especificado por el usuario, la generación de archivos que contengan listados de *URLs* a partir de una página web, un archivo *HTML* como los exportados por los principales navegadores o la generación de un listado a partir de los resultados obtenidos en una búsqueda por palabras clave en *Yahoo*.

En conclusión, el programa cumple los requisitos que se enunciaron para su desarrollo, realizando sin problemas los objetivos para los que fue creado (apartado 3.2.1). Sin embargo, es importante comentar los posibles problemas que puede encontrar ocasionalmente. En primer lugar, el programa trabaja en un sistema abierto y cambiante como es la web en el que existen una gran cantidad de factores difícilmente controlables y previsibles por parte del programa; el cambio de estructuras, la implementación de nuevas tecnologías o la evolución de las actuales, pueden afectar a su correcto funcionamiento. Por otro lado, la incorrecta implementación de la microdata en ciertas páginas web, así como su uso fuera del estándar enunciado por *schema.org* [19], puede llevar a que en ocasiones el programa solo pueda descargar parte de la información descrita en éstas.

5.2. Desarrollos futuros

El programa desarrollado en este proyecto, pretende ser una primera aproximación al posible uso de la información en forma de microdata que se dará en el futuro. La localización y extracción de la microdata implica la obtención de información de gran riqueza semántica que puede ser usada en diversos ámbitos, desde la recopilación de información sobre un sujeto o tema concreto, a su uso como posible entrada de información en distintos programas informáticos e, incluso, como se ha visto en este proyecto (apartado 1.1), la experimentación y desarrollo de sistemas robóticos.

El programa ha sido desarrollado con el objetivo de localizar y extraer imágenes con microdata. Sin embargo, existe una gran variedad de clases dentro de su vocabulario actual que pueden resultar interesantes de localizar y extraer en un futuro. La modularidad del programa permite que éste pueda ser ampliado, sin la necesidad de crear una nueva estructura, a la localización y extracción de otras clases de microdata desde otros elementos multimedia, como vídeos o audio, a la información referente a personas o empresas e, incluso, la recopilación de información médica, campo en el que se está usando cada vez más este tipo de marcado.

Por otro lado, puede mejorarse la información obtenida por el programa mediante la captura y extracción de otros lenguajes semánticos. Uno de sus posibles desarrollos futuros sería la localización, procesamiento y extracción de información presente en tripletas *RDF*, o a partir de otros tipos de metadatos tales como *RDFa* o microformatos. El aumento de la presencia de tripletas *RDF* dentro del archivo binario de imágenes puede suponer el siguiente paso lógico en la evolución del programa. Orientar el desarrollo futuro del programa a la extracción de estos lenguajes semánticos supone un reto importante y difícil, pero a cambio puede obtenerse una potente herramienta con una gran flexibilidad.

Otro de los posibles desarrollos futuros del programa puede pasar por que éste manipule la información extraída de la microdata. Para ello debería realizarse el programa un desarrollo orientado a objetos creados a partir de la estructura de clases del vocabulario de la microdata. Esta implementación permitiría la comparación y manipulación de distintos datos e, incluso, la creación de nueva información inferida a partir de la extraída.

Además, en un futuro puede desarrollarse con mayor profundidad la interpretación y manipulación de los datos *Exif*. Si se relaciona la microdata con los metadatos *Exif* pueden inferirse nuevas interpretaciones generando nueva información.

La naturaleza de *araña web* del programa permite un amplio abanico de posibles usos y desarrollos del mismo. El programa puede seguir usándose para la extracción de información semántica o adaptarse a cualquier de los múltiples usos actuales y futuros de este tipo de *Web bots*, desde el mantenimiento de dominios web a la recopilación de todo tipo de información: los precios de la competencia en un mercado determinado, las conductas y comportamientos de los usuarios y empresas en redes sociales, etc. En general, para cualquier campo en el que sea útil la extracción de

información a partir de la web, el uso de un programa *crawler* es en la mayoría de las ocasiones la mejor solución.

Uno de los desarrollos futuros más interesantes del programa actual pasa por la mejora y perfeccionamiento del *crawler* realizado. Como se ha comentado anteriormente, la creación e implementación de una *araña web* supone un reto relativamente sencillo de alcanzar. La dificultad llega a la hora de conseguir un buen rendimiento y eficacia del mismo. La creación de nuevas políticas de navegación y mejoras de las existentes es un posible desarrollo futuro que puede permitir al programa cosas como ser capaz de descartar páginas con bajas probabilidades de presencia de microdata, así como la selección de rutas más complejas y eficaces a través de la web.

Como se puede observar, el programa *Onewebmicrodata* dispone de un amplio abanico de posibilidades futuras, tanto por su condición de *crawler*, como por su posibilidad de extraer y manipular información semántica como la presente en la microdata, y que supone un avance significativo en la expansión de los datos semánticos en la web.

Apéndice A

Manual de uso

A.1. Introducción

Este es manual de uso para el programa *Onewebmicrodata* realizado por Julián Caro Linares como parte central del proyecto *Onewebmicrodata: Araña web extractora de Metadatos de Microdata y Exif* .

La función principal del programa es la búsqueda de imágenes con datos semánticos en formato de microdata a través de la web así como su extracción junto a la imagen que describen.

A.2. Compilación y ejecución del programa

El programa ha sido desarrollado usando el lenguaje *C++* y mediante el uso de *QT creator*.

Para realizar la compilación extraiga la carpeta raíz en un directorio cualquiera. Existen dos opciones para compilar el programa. Mediante el uso del *IDE QT creator* o usando *CMake*:

- **QT creator:** Para compilar el programa mediante *QT creator*:

  1. Abra el programa *QT creator*.
  2. File: Open File or project.
  3. En la carpeta raíz del programa, abra el archivo *.pro*.
  4. *QT creator* le mostrará opciones de configuración. Configure sus preferencias y pulse *Configure Project*.
  5. A continuación se le mostrará los archivos del programa para su edición. Para compilar sobre la barra izquierda de *QT* pulse el ícono del martillo *Build Project* el programa se compilara y estará listo para su ejecución.[1]

[1] Dependiendo de la versión de *QT* puede que la opción *RUN* de ejecutar el programa dentro de *QT* no funcione correctamente. Se recomienda siempre ejecutar el programa fuera de *QT creator* mediante una terminal.

Para ejecutar el programa diríjase a la carpeta que *QT* ha creado con el programa compilado. Desde una terminal escriba:

```
gcc  -o hello hello.c
```

- **CMake:** Para compilar mediante el uso de *CMake:*

  1. Abra en una terminal la carpeta raíz del programa.
  2. Una vez dentro de la carpeta raíz del programa se recomienda crear un directorio dentro de esta para la compilación del programa, por ejemplo *build*. Para ello ejecute el comando *$ mkdir build*.
  3. Sitúese en la carpeta donde desea que se realice la compilación del programa
  4. Ejecute el comando $ cmake *carpetaraizdelprograma*. En el caso de que haya creado la carpeta *build* comentada anteriormente el comando sería $ cmake ..
  5. *CMake* creará los archivos necesarios para la compilación. Si todo ha ido bien ejecute el comando $ make.

  Si la compilación ha resultado un éxito tendrá un nuevo archivo ejecutable denominado *Onewebmicrodata* que será el programa compilado. Para ejecutarlo escriba $ *./Onewebmicrodata* .

NOTA: El programa depende de la librería externa *libcurl*, si la librería por alguna razón no está instalada en su ordenador, el compilador le avisara de ello, puede descargar la librería ejecutando el comando:

```
sudo apt-get install libcurl-dev
```

A.3. Opciones del programa y configuración

Una vez abierto el programa se encontrará con el menú principal (figura A.1). En él se enumeran las distintas opciones disponibles. Seleccione la que desee escribiendo su número y pulsando *intro*.

Antes de describir cada opción, existen una serie de parámetros de navegación comunes cuyo significado es preciso que conozca:

- **Nº de iteraciones:** Determina la profundidad de navegación, o en otras palabras, el número de veces que el programa saltará a otra página desde el sitio web inicial. Una vez el programa ha analizado el primer sitio web analizará las páginas web vinculados a este y luego saltará de forma pseudoaleatoria a uno de ellos repitiendo el proceso de análisis del sitio web, análisis

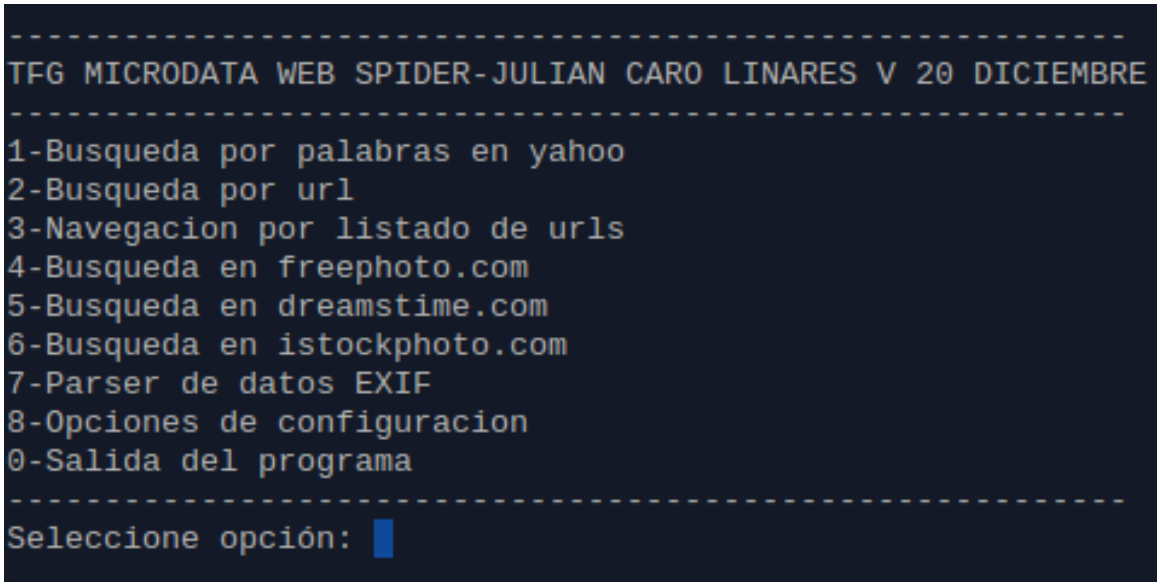

Figura A.1: Menú principal del programa. En él se enumeran las distintas opciones de ejecución. Para ejecutar una opción introduzca el número de la opción seleccionada y pulse *intro*.

de las páginas web vinculadas y salto a una nueva página web sucesivamente dependiendo del número de iteraciones que hayamos seleccionado.

Es **importante** destacar que sí seleccionamos solo una iteración el programa analizará solo la página inicial, **sin analizar las páginas vinculadas a esta**. Para realizar un análisis de la primera página más sus páginas referenciadas deberemos escribir al menos dos iteraciones.

- **Porcentaje de análisis:** Índica al programa el porcentaje de análisis que deseamos analizar sobre el total de páginas referenciadas en cada sitio web visitado. Si por ejemplo tenemos una página web inicial que vincula a 10 páginas web si seleccionamos un porcentaje del 50 % solo se analizarán 5 de estás de una forma pseudoaleatoría. Este parámetro puede resultar de gran utilidad en ocasiones en las que el programa navega por páginas web con un elevado número de links referenciados y se desea un análisis más superficial. Al finalizar el análisis de dichas páginas, el programa saltará a una de las que se han analizado comenzando otra iteración.

- **Incluir o no links locales:** Donde se entiende por links locales a los links parciales a páginas que normalmente son del mismo dominio. Para diferenciar entre un link parcial y uno absoluto observe las siguientes *URLs*:

  http://www.uc3m.es/Inicio

  $/Vida\_Universitaria$

  Mientras que el primero es una *URL* totalmente funcional, la segunda url es un link parcial

que apunta a otra página del mismo dominio. Si usted selecciona que desea incluir los links locales este tipo de rutas parciales se reconstruirán resultando el caso anterior:

$$http://www.uc3m.es/Vida\_Universitaria$$

Incluir los links locales en la búsqueda implica un mayor número de páginas refenciadas a analizar por página pero a su vez una mayor probabilidad de que el programa analice más páginas del mismo dominio antes de conseguir iterar a otro dominio distinto. Normalmente es recomendable incluir los links locales en nuestra búsqueda. Sin embargo, en casos en los que el número de links locales es mucho mayor al de link externos puede ser recomendable desactivar esta opción.

A continuación se enumeran y describen cada una de las opciones seleccionables:

1. **Busqueda por palabras en Yahoo:** Permite mediante la introducción de una serie de palabras clave empezar a realizar una búsqueda de microdata según los resultados ofrecidos por *Yahoo*. En primer lugar se le pedirá que escriba las palabras clave, escriba una o varias separadas por espacios y pulse *intro*. A continuación se le pedirá el número de páginas de resultados que desea analizar, cada página ofrece diez resultados, por lo que si por ejemplo se analizan 10 páginas, el número de links ofrecidos por *yahoo* será de 100. Acto seguido se le pedirá el número de iteraciones, el porcentaje de análisis y si desea incluir o no los links parciales obtenidos de cada página web.

   Esta opción puede serle de utilidad cuando desea comenzar una búsqueda determinada pero desconoce direcciones webs concretas que puedan tener dicha información.

2. **Búsqueda por url:** Cuando usted conozca un dominio determinado desde el que desea comenzar una búsqueda utilice esta opción. Se le pedirá que introduzca la *URL* deseada así como el número de iteraciones, porcentaje de análisis y si desea o no incluir los links locales en la búsqueda.

   Es importante que escriba la dirección *URL* correctamente. El programa acepta varias formas para escribir la dirección:

   http://www.uc3m.es/Inicio

   www.uc3m.es/Inicio

   uc3m.es/Inicio

   Sin embargo es recomendable que use siempre la primera escribiendo la *URL* completa. Una forma sencilla es que copie y pegue la *URL* cuando se la solicite el programa (normalmente

deberá presionar *control+shift+v)*. Recuerde asegurarse siempre de comprobar que la *URL* que introduce es una *URL* valida y no una página caída o dirección errónea. En caso contrario el programa intentará analizar dicha dirección web sin conseguir iterar a una nueva página realizando el número de iteraciones solicitadas sin éxito y devolviéndole de nuevo al menú principal.

3. **Navegación por listado de urls:** Esta opción es una ampliación de la opción de búsqueda por *URL* en la que dada una lista de *URLs* a analizar el programa realizará la exploración de cada uno de sus links de la lista teniendo en cuenta los parámetros habituales de iteraciones o profundidad, porcentaje de análisis e inclusión o no de links locales. Es decir, el programa leerá uno de los links del archivo, realizará el número de iteraciones deseadas y al terminar leerá el siguiente link repitiendo el proceso hasta completar la lista.

   El formato que deben tener dichos archivos es el de un *archivo con una URL por línea*. En la figura A.2 puede ver un ejemplo sencillo de como debe ser un archivo tipo.

Figura A.2: Ejemplo de listado de *URLs* que se desea analizar. Debe escribirse una sola *URL* por línea.

   Al seleccionar dicha opción el programa le solicitará el nombre o ruta del archivo que contiene la lista de los links. Sí el archivo se encuentra en la carpeta raíz del programa puede escribir su nombre directamente. Si no deberá escribir *su ruta completa*.

   A continuación se le pedirá que introduzca el nombre de una carpeta para dicha búsqueda. Dicha carpeta se creará en caso de que ya no existiera en *./databasemicrodata/nombrecarpeta*

   Después se le solicitará el número de links por el que desea comenzar la búsqueda. Esta opción

puede ser útil si ya ha analizado una serie determinada de links de dicho archivo y desea continuar donde lo dejo. En caso de que desee analizar la lista completa, escriba uno.

Por último se le pedirá los parámetros de navegación: número de iteraciones, porcentaje de análisis e inclusión o no de links locales.

En este modo de navegación se creará un archivo con el nombre de *results+nombrearchivo* en la misma ruta del archivo a cargar. Este archivo de resultados le informará del número de iteraciones, páginas web analizadas, así como si se ha encontrado o no potencialmente microdata en cada link analizado y en el total del análisis (figura A.3).

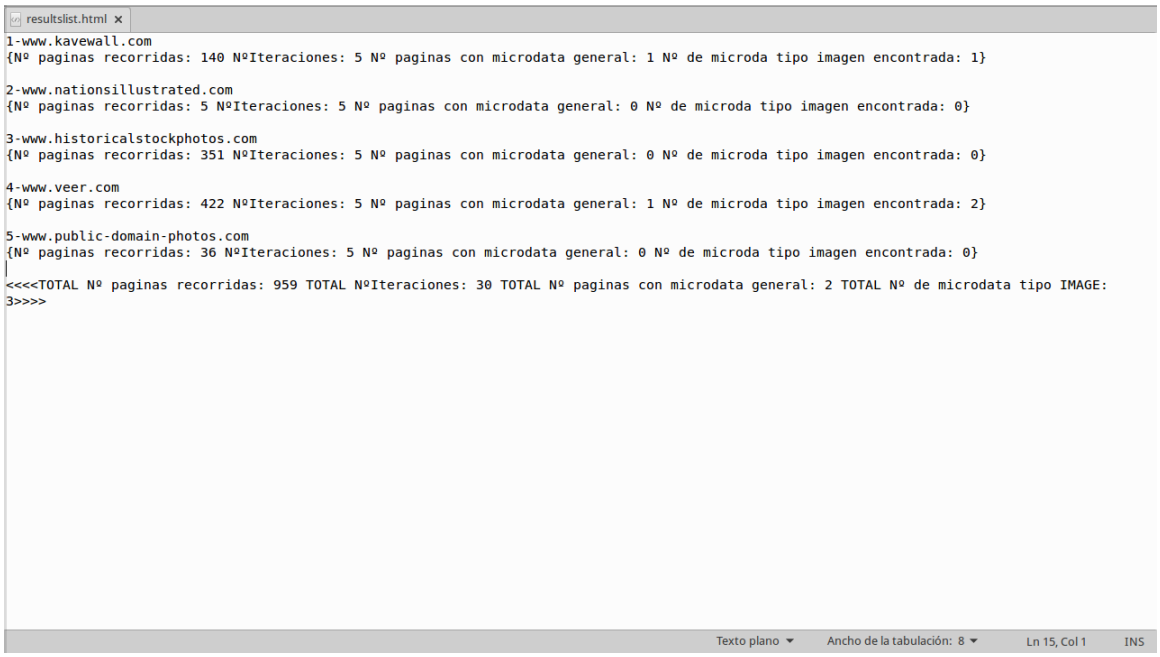

Figura A.3: Ejemplo de archivo de resultados tras realizar una búsqueda usando la opción *Navegación por listado de urls*

NOTA: Realizar listados de *URLs* puede llegar a ser tedioso y exigirle mucho tiempo. Más adelante en el apartado *opciones de configuración* se describirán algunas de las opciones que el programa posee para realizar archivos de listados de *URLs* de forma automática.

4. **Búsqueda en freefoto.com:** Esta opción permite realizar una búsqueda de microdata mediante palabras clave en el banco de imágenes de freefoto.com. En primer lugar se le pedirá que introduzca una o más palabras claves. A continuación el número de iteraciones deseadas. Tenga en cuenta que en este modo de exploración no se iterará de una forma pseudoaletoría si no que el número de iteraciones corresponde al número de páginas de resultados a analizar según las palabras clave introducidas. A mayor número de iteraciones, mayor número de páginas de resultados se analizarán obteniendo un número mayor de imágenes con microdata.

Al finalizar la búsqueda podrá encontrar las imágenes con microdata obtenidas en la carpeta *dabasemicrodata/palabrasclaveintroducidas*.

5. **Búsqueda en dreamstime.com:** Su funcionamiento es idéntico al de la opción de *freefoto* siendo el número de iteraciones el número de páginas de resultados a analizar.

6. **Busqueda en istockphoto.com:** Su funcionamiento es idéntico a los anteriores si bien puede observar un procedimiento de ejecución distinto. En este caso concreto de extracción de imágenes con información semántica se ha buscado optimizar el rendimiento al máximo realizándose la descarga de las imágenes encontradas solo al final del programa.

7. **Parser de datos Exif:** En todas las opciones ya descritas el programa realiza la extracción de los datos *Exif* de las imágenes encontradas de forma automática. Sin embargo puede resultarle útil realizar esta extracción para un grupo de imágenes de una carpeta específica de su ordenador. Para ello se le solicitará la ruta a analizar, si la ruta se encuentra dentro de la raíz del programa puede escribir la ruta parcial, en caso contrario deberá escribir la ruta completa.

   Al finalizar encontrará en la ruta introducida las fotos analizadas junto con un documento *txt* de su mismo nombre con la información *Exif* extraída así como con una interpretación automática de dichos datos.

   NOTA: Puede que observe por pantalla como el extractor de datos *Exif* arroja el mensaje: *Error parsing EXIF: code 1983*. Este mensaje es normal y propio de la librería de datos *Exif* y significa que en dicha imagen no se han encontrado datos a extraer.

8. **Opciones de configuración:** Por último el programa ofrece varias opciones de configuración que definen el comportamiento de la navegación de este por la web así como añade otras pequeñas funcionalidades:

   1. **Activar/Desactivar búsqueda de imágenes por palabras clave:** El programa dispone de una función adicional consistente en la descarga de imágenes sin microdata que contengan, bien en su descripción o en su nombre, una palabra o frase clave escrita. Dicha opción está desactivada por defecto. Si seleccionamos está opción la activaremos para la sesión actual del programa y se nos solicitará que introduzcamos la palabra clave deseada. A continuación desde el menú principal podremos seleccionar la opción que deseemos para explorar la Web.

      Podemos encontrar las imágenes introducidas dentro de: *databasemicrodata/carpetadeexploracion/imagenessinmicrodata/palabrasclave*

   2. **Opciones de historial de links ya analizados:** El programa **por defecto tiene activada la opción de historial**. Para facilitar la tarea de buscar microdata en la Web

el programa va generando un archivo de historial con todas las *URL* que ha visitado. De está forma cuando el programa se encuentra de nuevo con una página ya visitada, no la descarga y pasa a la siguiente dirección. Cuando el historial se hace demasiado grande o por otros motivos queremos desactivarlo pueden ser de utilidad las siguientes opciones:

1) **Activar/Desactivar uso de historial en navegación:** Estando activado por defecto, permite activar o desactivar el uso del historial para la sesión actual del programa.
2) **Usar un historial nuevo para esta sesión-El historial actual sera borrado: Borra el historial que pudiera existir** y usa un nuevo historial para esta sesión. Al final del programa el historial de la sesión actual será también borrado.
3) **Usar historial permanente:** En el caso de que haya seleccionado previamente la opción de historial nuevo para esta sesión, puede que finamente desee que el historial de la sesión no sea borrado. Ejecute esta opción antes de abandonar el programa para que este no se borre. La opción de historial permanente, activada por defecto, crea un historial que se guarda en cada sesión incrementándose con el uso del programa.
4) **Importar historial:** El historial no es más que un archivo de texto **con una url por línea**. Si desea importar otros historiales o crear el suyo propio y usarlo en el programa, puede usar esta opción para importar un archivo y crear un nuevo historial de navegación. Para ello seleccione esta opción e introduzca la ruta o nombre del archivo deseado. **Recuerde que si importa un historial, perderá el anterior.**
5) **Exportar historial:** Si desea guardar su archivo de historial fuera del programa ejecute esta opción. Se le pedirá que introduzca el nombre o la ruta más el nombre que desea que tenga el nuevo archivo.
6) **Eliminar historial:** Elimina el archivo de historial actual.

3. **Obtener listado de links a partir de url/palabras clave o archivo html:** Esta opción le permite obtener un listado de *URLs* en un archivo que puede ser usado para la opción de *Navegación por listado de urls*. Para ello dispone de dos opciones:

   1) **Introducir URL:** Introduzca una *URL* a una dirección web valida, el programa descargará la página y extraerá los links a los que apunta. Se le pedirá ademas el nombre de la carpeta donde desea encontrar los archivos que se generen.
      NOTA: Esta opción transforma e incluye los links locales y parciales a *URLs* validas por defecto.
   2) **Introducir ruta a archivo html:** Introduzca el nombre o la ruta del archivo **HTML** del que desea que el programa analice y extraiga sus links. A continuación se le pedirá un nombre para el nuevo archivo.

3) **Resultados de busqueda en Yahoo:** Le permite guardar los resultados arrojados por una búsqueda por palabras clave en *yahoo* a un archivo *HTML*. Ésta opción puede resultar muy útil en el caso de que desee generar un listado sobre un tema del que desconoce a priori direcciones web.

4. **Activar/Desactivar análisis de microdata por pagina:** Por defecto cuando el programa encuentra una serie de microdata en una página web, analiza, descarga y extrae la foto y la información correspondiente en ese mismo momento. Esto evita que pueda haber una perdida de datos en el caso de que el programa se interrumpa por cualquier motivo. Si por el contrario desea que dicho análisis se ejecute solo al final de las iteraciones ejecute esta opción para activar o desactivar el análisis por página.

A.4. Un ejemplo de uso

A continuación vamos a realizar un ejemplo de uso del programa. Buscaremos imágenes con microdata en la página *http://www.theguardian.com/uk*. El diario *theguardian.com* usa un poco de datos con microdata para definir las fotografías presentes en sus artículos. Deseamos obtener unas cuantas de estas fotografías.

Figura A.4: Portada del periódico *theguardian.com* en su versión inglesa.

Para ello seguimos los siguientes pasos:

1. Ejecutamos el programa, como tenemos la dirección *URL* donde nos interesa comenzar la búsqueda, seleccionamos la opción dos: *Busqueda por URL*.

2. Introducimos la *URL: http://www.theguardian.com/uk*.

3. Como deseamos solo unas pocas imágenes con microdata y no una búsqueda exhaustiva, seleccionamos solo cinco iteraciones. Al ser una página de noticias con muchos links relacionados puede que el programa tarde un tiempo en realizar la exploración completa.

4. Seleccionamos un porcentaje de análisis del 100 % (escriba el número 100 y pulse *intro* sin introducir el carácter % de porcentaje) o el que deseemos y si queremos incluir o no los links locales. Como deseamos obtener microdata de este dominio web lo recomendable es incluirlos. Establecemos dichas opciones y presionamos *intro*.

El programa comenzará a explorar la Web en busca de *microdata* sucediéndose de forma rápida una serie de información por la pantalla de la terminal. Si vamos a la carpeta: *raizprograma/databasemicrodata/theguar uk* podemos ver como van apareciendo las imágenes, así como su microdata extraída en un *txt* del mismo nombre, conforme el *crawler* detecta, extrae, analiza y descarga dicha información. En el caso de que no haya encontrado imágenes con microdata. La carpeta permanecerá vacía. Al final de la exploración su carpeta debería contener algo parecido a esto:

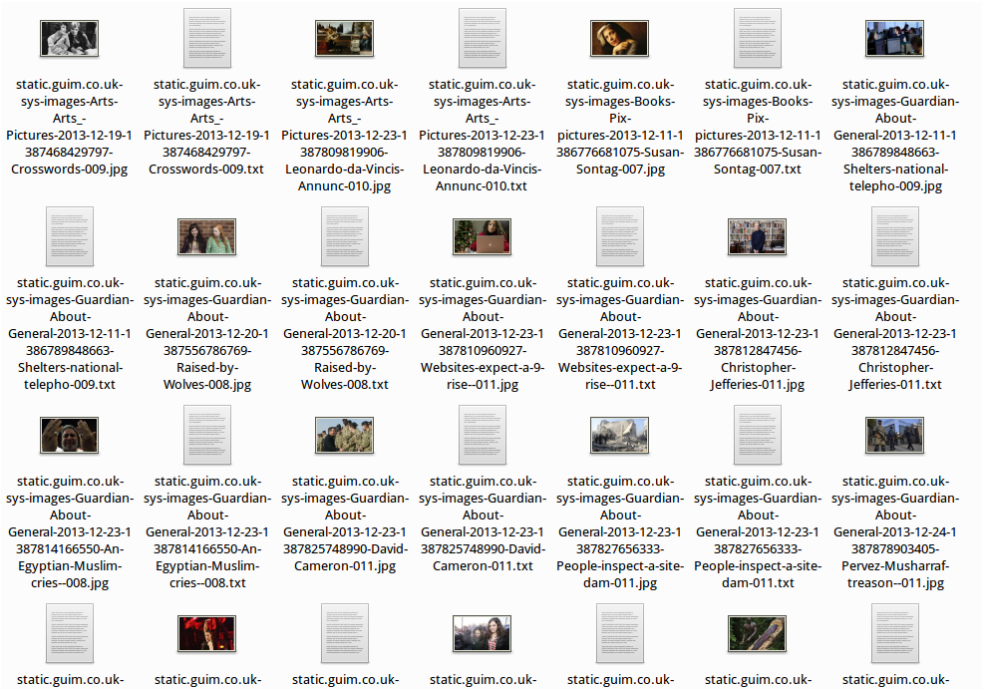

Figura A.5: Resultados del ejemplo extraídos de la página *http://www.theguardian.com/uk*.

Recuerde que durante la exploración el programa ha guardado por defecto las *URLs* ya visitadas. Si a continuación realiza una exploración idéntica el programa no descargará las páginas visitadas analizando solo nuevas páginas lo que le permitirá obtener nueva información sin necesidad de recorrer de nuevo dichas páginas. Puede cambiar el comportamiento del historial en: *Opciones de configuración*.

NOTA IMPORTANTE: Al final de la exploración aparecerán en la terminal una serie de informaciones como el número de páginas recorridas, el número de iteraciones efectuadas, así como el número de páginas con microdata, no necesariamente de tipo imagen, encontradas en la web y el número de etiquetas con microdata tipo imagen que se han hallado. Es posible que estás cifras varíen un poco respecto a lo que se espera. En ocasiones el programa se encuentra con páginas vacías o redirecciónes que le obligan a realizar más iteraciones de las deseadas. Por otra parte, el número de microdata tipo *image* encontrada puede ser mayor al número de imágenes finales con microdata descargadas. Esto se debe a que durante el análisis pueden encontrarse imágenes duplicadas o ya existentes o información de microdata que tras es análisis no es valida.

A.5. Recomendaciones y posibles problemas

A continuación se exponen una serie de recomendaciones, trucos y posibles problemas que pueden serle útiles a la hora de utilizar el programa:

- Copie y pegue las url que desea introducir en el programa en vez de escribirlas por teclado. Ahorrará tiempo y evitará errores en su escritura. Para copiar y pegar en un terminal deberá usar los comandos *control+shift+c* y *control+shift+v* respectivamente.

- Utilice la opción de *búsqueda por listado de urls* para automatizar al máximo el programa. Cree sus propios listados de *URLs* que considere útil analizar o use la opción *Obtener listado de links a partir de url o archivo* para obtener listados de forma sencilla. Puede exportar sus marcadores a un archivo *HTML* en la mayoría de los navegadores web existentes. Use el archivo exportado junto a la opción anterior para obtener un listado de las *URL* de sus marcadores. También existen numerosos servicios webs como Delicious.com que le permitirán exportar sus marcadores a un documento *HTML*.

- Tenga cuidado con el número de iteraciones introducido. Algunas páginas poseen una gran cantidad de links vinculados a esta. Si introduce un número de iteraciones muy elevado el programa puede tardar mucho tiempo en terminar la exploración. Si desea tener el programa funcionando mucho tiempo, es más recomendable que use la opción de *busqueda por listado de url* con un número de iteraciones por *URL* más razonable. Esta opción le permitirá continuar el análisis por el número de link deseado en caso que sea necesario.

- Si conoce el funcionamiento de la *web request* de un dominio en concreto, puede usarlo introduciéndola en el *caso 2: Búsqueda por url* para mejorar sus resultados de búsqueda.

- Use el historial con habilidad. Aunque la lectura y escritura del historial no supone una disminución del rendimiento drástica para el programa, llegado un momento al tener muchos links almacenados en el archivo puede que el rendimiento de este disminuya. Si es el caso puede utilizar algunas de las opciones de *Opciones de historial de links ya analizados* para eliminar, desactivar temporalmente o exportar entre otras el historial y obtener mayor rendimiento durante su exploración.

- En caso de que, por cualquier motivo, el programa haya sido interrumpido, si éste había encontrado microdata pero no llegó a descargarla le preguntará si desea recuperarla al iniciar de nuevo el programa. En el caso de que decida recuperarla, ésta se almacenará en la ruta *databasemicrodata/recoveryfiles*.

- Puede que algunas páginas web tarden más o menos tiempo en descargarse, sea paciente. La velocidad del programa dependerá sobre todo de su conexión a Internet así como del dominio donde se está descargando cada página. En el caso de que una página tarde más de diez minutos en realizar la descarga, el programa automáticamente cerrará la conexión y pasará a la siguiente página.

Bibliografía

[1] J. G. Victores, S. Morante, A. Jardon, and C. Balaguer, "Towards robot imagination through object feature inference," *IROS IEEE/RSJ International Conference on Intelligent Robots and Systems. Tokyo. Japan*, 2013.

[2] J. Heldler, T. Berners-Lee, and E. Miller, "Agent technology on the internet. 3. integrating applications on the semantic web.," *Journal of the Institute of Electrical Engineers of Japan*, vol. 122, no. 10, pp. 676–680, 2002.

[3] V. Bush, "As we may think," *The Atlantic Monthly*, July 1945.

[4] J. Conklin, "Hypertext: An introduction and survey," 1987.

[5] D. Goodman, *Complete HyperCard 2.0 Handbook*. Random House Inc., 1990.

[6] J. Gilles and R. Caillau, *How the Web was born: The story of the World Wide Web*. Oxford University Press, 2000.

[7] "World wide web, the first website of the history." `http://www.w3.org/History/19921103-hypertext/hypertext/WWW/TheProject.html` (último acceso: Febrero 2014), Agosto 1991.

[8] "Extensible markup language (xml)." `http://www.w3.org/TR/WD-xml-961114.html` (último acceso: Febrero 2014), Noviembre 1996.

[9] T. Berners-Lee, "W3C talks 1994." `http://www.w3.org/Talks/WWW94Tim/` (último acceso: Febrero 2014), Septiembre 1994.

[10] T. Berners-Lee, "Layer cake." `http://www.w3.org/2001/09/06-ecdl/slide17-0.html` (último acceso: Febrero 2014), 2000.

[11] R. Studer, V.Benjamins, Richard, and D. Fensel, "Knowledge engineering: principles and methods," *Data & knowledge engineering*, vol. 25, no. 1, pp. 161–197, 1998.

[12] T. Berners-Lee, "W3c semantic web activity." `http://www.w3.org/2001/sw/` (último acceso: Febrero 2014), 2006.

[13] R. García and R. Gil, "Publishing xbrl as linked open data," in *CEUR Workshop Proceedings*, vol. 538, Citeseer, 2009.

[14] K. A.Haffner, ed., *Semantic Web-Standars, Tools and Ontologies*. Nova, 2010.

[15] N. C. Institute, "Cancer genome atlas." `http://cancergenome.nih.gov/` (último acceso: Febrero 2014), Diciembre 2013.

[16] H. F. Deus, D. F. Veiga, P. R. Freire, J. N. Weinstein, G. B. Mills, and J. S. Almeida, "Exposing the cancer genome atlas as a sparql endpoint," vol. 43, no. 6, pp. 998–1008, 2010.

[17] T. Berners-Lee, "The year open data went worldwide." `http://www.ted.com/talks/tim_berners_lee_the_year_open_data_went_worldwide.html` (último acceso: Febrero 2014), Febrero 2010.

[18] "Facebook open graph api." `https://developers.facebook.com/docs/opengraph/` (último acceso: Febrero 2014), Diciembre 2013.

[19] "Schema.org." `http://schema.org/` (último acceso: Febrero 2014), Diciembre 2013.

[20] "Schema.org-person." `http://schema.org/Person` (último acceso: Febrero 2014), Diciembre 2013.

[21] G. Inc, "Google structured data testing tool." `http://www.google.es/webmasters/tools/richsnippets` (último acceso: Febrero 2014), Diciembre 2013.

[22] L. Clark, "Drupal microdata." `https://drupal.org/project/microdata` (último acceso: Febrero 2014), Junio 2011.

[23] G. Inc, "Google recipes view." `http://www.google.com/insidesearch/features/recipes/` (último acceso: Febrero 2014), 2013.

[24] I. Inc, "Bot traffic report 2013." `http://www.incapsula.com/the-incapsula-blog/item/820-bot-traffic-report-2013` (último acceso: Febrero 2014), Diciembre 2013.

[25] "Slug: A semantic web crawler." `http://www.ldodds.com/projects/slug/` (último acceso: Febrero 2014), Diciembre 2013.

[26] "Schema.org-imageobject." `http://schema.org/ImageObject` (último acceso: Febrero 2014), Diciembre 2013.

[27] "Schema.org-photograph." `http://schema.org/Photograph` (último acceso: Febrero 2014), Diciembre 2013.

[28] "Libcurl-the multiprotocol file transfer library." `http://curl.haxx.se/libcurl/` (último acceso: Febrero 2014), Diciembre 2013.

[29] H. Young, "Libghttp: Gnome http client library." `http://sourceforge.net/projects/cygnome/files/Core%20Libraries/libghttp-1.0.9/` (último acceso: Febrero 2014), Marzo 2009.

[30] H. F. Nielsen, T. Berners-Lee, and J.-F. Groff, "Libwww - the w3c protocol library." `http://www.w3.org/Library/` (último acceso: Febrero 2014), Octubre 2003.

[31] A. E. Lee Thomason, Yves Berquin, "Tinyxml." `http://www.grinninglizard.com/tinyxml/` (último acceso: Febrero 2014), Diciembre 2013.

[32] D. Veillard, "Libxml: The xml c parser and toolkit of gnome." `http://www.xmlsoft.org/` (último acceso: Febrero 2014), Diciembre 2013.

[33] "Irrxml." `http://www.ambiera.com/irrxml/` (último acceso: Febrero 2014), Diciembre 2013.

[34] L. Thomason, Y. Berquin, and A. Ellerton, "Tinyxml2." `https://github.com/leethomason/tinyxml2` (último acceso: Febrero 2014), Diciembre 2013.

[35] M. Lahiri, "Easyexif: Tiny iso-compliant c++ exif parsinf library." `https://code.google.com/p/easyexif/` (último acceso: Febrero 2014), Diciembre 2013.

[36] A. Huggel, "Exiv2: Image metadata library and tools." `http://www.exiv2.org/` (último acceso: Febrero 2014), Diciembre 2013.

[37] "The libexif c exif library." `http://libexif.sourceforge.net/` (último acceso: Febrero 2014), Diciembre 2013.

[38] "Yahoo search engine." `http://search.yahoo.com/` (último acceso: Febrero 2014), Enero 2014.

[39] M. J. Langford and M. Langford, *La fotografia paso a paso/The Picture Step by Step: Un Curso Completo*. Ediciones AKAL, 1990.

[40] J. Hedgecoe, *Manual de técnica fotográfica*. Ediciones AKAL, 1992.

[41] "Sindice: The semantic web index." `http://sindice.com/search?q=&nq=&fq=&interface=guru&facet.field=domain` (último acceso: Febrero 2014), Febrero 2014.

[42] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data-the story so far," *International Journal on Semantic Web and Information Systems (IJSWIS)*, 2009.

[43] G. Klyne, J. J. Carroll, and B. McBride, "Resource description framework (rdf): Concepts and abstract syntax," *W3C recommendation*, vol. 10, 2004.

[44] T. Berners-Lee, "Linked data-design issues (2006)," *URL http://www. w3. org/DesignIssues/-LinkedData. html*, 2011.

[45] C. Castillo, "Effective web crawling," 2004.

[46] J. E. Labra Gayo, *Web Semántica: Comprendiendo el cambio hacia la Web 3.0.* Pocket Innova, netbiblo, 2011.

[47] J. A. P. Sánchez, *Tecnologías de la Web Semántica.* El profesional de la información, UOC, 2011.

[48] D. Allemang and J. Hendler, *Semantic web for the working ontologist: effective modeling in RDFS and OWL.* Access Online via Elsevier, 2011.

[49] M. Schrenk, *Webbots, spiders, and screen scrapers.* No Starch Press, 2012.

[50] K. Ray, "Web 3.0. a story about the semantic web." `https://vimeo.com/11529540#at=0` (último acceso: Febrero 2014), Mayo 2010.

[51] K. Kelly, "Kevin kelly on the next 5000 days of the web." `http://www.ted.com/talks/kevin_kelly_on_the_next_5_000_days_of_the_web.html` (último acceso: Febrero 2014), Diciembre 2007.

[52] I. Herman, "Web ontology language (owl)." `http://www.w3.org/2004/OWL/` (último acceso: Febrero 2014), Octubre 2007.

[53] W3C, "Owl 2 web ontology language document overview (second edition)." `http://www.w3.org/TR/owl2-overview/` (último acceso: Febrero 2014), Diciembre 2012.

[54] P. Matthew Horridge and F. Patel-Schneider, "Owl 2 web ontology language: Manchester syntax." `http://www.w3.org/TR/2008/WD-owl2-manchester-syntax-20081202/#Introduction` (último acceso: Febrero 2014), Diciembre 2008.

[55] W3C, "Sparql 1.1 overview." `http://www.w3.org/TR/sparql11-overview/` (último acceso: Febrero 2014), Marzo 2013.

[56] W3C, "Sparqlendpoints." `http://www.w3.org/wiki/SparqlEndpoints` (último acceso: Febrero 2014), ENERO 2013.

[57] U. of Mannheim, "Web data commons extraction report - august 2012 corpus." `http://webdatacommons.org/2012-08/stats/stats.html` (último acceso: Febrero 2014), Agosto 2012.

[58] "Microformats.org." `http://microformats.org/wiki/Main_Page` (último acceso: Febrero 2014), Agosto 2013.

[59] Google, "Herramientas para webmasters de google-fragmentos enriquecidos: personas." `https://support.google.com/webmasters/answer/146646` (último acceso: Febrero 2014), 2013.

[60] I. Herman, B. Adida, M. Sporny, and M. Birbeck, "Rdfa 1.1 primer - second edition." `http://www.w3.org/TR/rdfa-primer/` (último acceso: Febrero 2014), Agosto 2013.